£3.20

.EXE

The Software Developers' Magazine

APRIL 1992 VOL 6 ISSUE 10

They can't buy it, but we've got it - .EXE cuts the hype out of OOP.

Is OOP methodology trailing the technology? We have a Software Engineering perspective.

The inside story on C7 development - Frank views from Microsoft's Dave Weil.

A neat add-on for Borland's Turbo Vision.

Plus code, code and more code: 3D Graphics, safer classes, XMS & Windows.

BORLAND

The Leader in Object-Oriented Programming

Now in its third generation! Tested, proven, reliable!

Professional C programmers are choosing object-oriented C++ because they know it increases productivity. And the C++ they're choosing is from Borland. With more than 500,000 copies of C++ shipped, Borland sets the standard.

Since 1983, Borland has consistently provided professional software development tools to programmers around the world. Today, Borland is the acknowledged leader in Object-Oriented Programming (OOP). OOP gives you amazing code reusability, extensibility and easier maintenance because applications are built on a basis of tested, reliable code.

Borland C++ 3.0. C++ you can rely on!

Borland® C++ is the choice of professional C *and* C++ programmers for Windows *and* DOS application development. The third-generation C++ compiler fully supports the AT&T CFRONT 3.0 standard C++, including templates! And the powerful C compiler fully implements the ANSI C standard.

With both compilers included you can program in C today and move to C++ when you're ready.

A host of powerful features in one package!

Borland C++ comes with an array of powerful yet easy-to-use features demanded by today's professional programmers. The **Windows-hosted IDE** provides a superior software development environment. Now you create, edit, compile and run Windows applications all from within Windows. All resulting in remarkable productivity.

C and C++ for Windows and DOS all in one package!

Only Borland C++ gives you the graphical **ObjectBrowser.**™ So you get a visual display of the relationships between objects, and can easily navigate through the classes, functions or variables in your code.

The **EasyWin**™ **Library** converts standard DOS applications to Windows just by recompiling. The **SpeedBar**™ makes Windows development astonishingly intuitive and fast, employing recognizable icons to represent your frequently used menu items.

Precompiled headers give you quick compilation and Borland C++'s DPMI support lets you compile huge applications fast. **Resource Workshop** allows you to visually create your Windows user interface without programming.

Quality tools that other compilers can't match

Borland C++ also comes complete with all the tools that the serious professional developer needs. Borland's award-winning **Turbo Debugger**® gives you intelligent, interactive debugging on a single monitor. The **Turbo Profiler**™ helps you spot bottlenecks in your code to streamline your application's performance. With the object-oriented **Turbo Assembler**® you can integrate your assembly language code into your applications quickly and easily.

Only Borland's C++ gives you Application Frameworks for DOS and Windows

Borland C++ & Application Frameworks slashes development time by giving you intuitive, ready-made user interfaces for Windows and DOS. Simply plug an application framework into your application or customize it any way you wish. High-level objects are ready to bolt on with just a few lines of code. Add an editor in just one line! Automatically inherit windows, dialogues, menus, mouse support and more.

Borland C++ includes

- C++ with templates, compatible with AT&T CFRONT 3.0
- ANSI C
- Global optimizer, including
  - Global register allocation
  - Local and global common sub-expressions
  - Induction variables
  - Loop and jump optimization
  - Register parameter passing
  - Ten other optimizations
- Fast compile speed
- Windows and DOS integrated development environments
- DPMI support for compiler and environments
- Visual ObjectBrowser that navigates through your code
- EasyWin Library for converting DOS programs to Windows
- Resource Workshop to create a complete range of Windows resources
- WinSight™ message tracking utility
- Turbo Debugger for DOS and Windows
- Turbo Profiler for DOS and Windows
- Object-oriented Turbo Assembler

Borland C++ & Application Frameworks also includes

- ObjectWindows for Windows
- Turbo Vision for DOS
- Complete source code for both Application Frameworks and the runtime library

1991 BYTE AWARD OF DISTINCTION

PC WEEK LABS TOP PRODUCTS OF 1991

J.D. POWER AND ASSOCIATES®
BORLAND
"Best application software in customer satisfaction in Small to Medium-sized and Large Businesses"

The **ObjectWindows**™ application framework streamlines Windows development. It automates initialization and gives your applications more functionality with less code.

ObjectWindows also features Borland Custom Controls that make your applications look great.

Also included is **Turbo Vision**,™ the application framework for DOS that gives you capabilities similar to ObjectWindows for character-based programs. Both come with full source code.

The critics' choice

"Borland C++ looks set to become the *de facto* development environment for Windows programmers."
—*Program Now*,
December, 1991

"There are many new features packed into this release . . . Microsoft® has a lot of catching up to do."
—*.EXE* magazine,
December 1991

Borland makes your transition from C to C++ easy!

To make your transition from C to C++ as smooth as possible, we've created *World of ObjectWindows for C++* training videotape. And now you can get it FREE (retail value £99.95)! This remarkable video training package comes complete with two videos, workbook and sample source code that you can plug right into your program! Taught by internationally recognized C++ experts, *World of ObjectWindows for C++* will give you an in-depth understanding of the ObjectWindows application frame-work for Windows. Combined with Borland C++ & Application Frameworks, there is no faster, more productive way to get up to speed on C++.

If you're serious about moving to C++, go with the leader, Borland.

CIRCLE NO. 812

FREE Video Training Package!

Borland is making your move to C++ as easy as possible with a special, limited-time offer. From now until 1 October 1992, you may redeem this official form for a free *World of ObjectWindows for C++* training package. Just purchase a copy of Borland C++ & Application Frameworks through your local dealer. Then send a copy of your invoice and your registration card, along with this official form, to:

Borland International (U.K.) Ltd.
Freepost RG1571
Twyford, Berkshire RG10 8BR

See your dealer today!

Or call Freefone 0800 212727

*1991 J.D. Power and Associates® Computer End User Satisfaction Study: Phase I & II Small to Medium-sized Businesses. (Responses from 1,784 business sites. with 1 to 499 employees) and Large Businesses (Responses from 1,094 business sites with 500 or more employees). The above J.D. Power and Associates rankings reflect only those companies above the industry average. Copyright © 1992 Borland International, Inc.
All rights reserved. All Borland products are trademarks of Borland International, Inc. BI 1514

BORLAND

Vol 6 Issue 10 *April 1992*

Editor:	Willie Watts
Features Editor:	Paul Kemp
Staff Writer:	Cliff Saran
Production Manager:	Katerina Adams
Design & Layout:	Mark English
Group Advertising Manager:	Sandra Inniss-Palmer
Assistant Advertising Manager:	Ed Butcher
Advertising Executive:	Marc Warren
Reader Services & Promotions:	Helena Adams
Administration/Subscriptions:	Rena Gibbs Sandy McDonnell
Cover Illustration:	Lynne Chapman

Apologies to Saki's shade

General

.EXE Magazine is independent and not affiliated to any vendor of hardware, software or services. It is published by Process Communications Ltd, 10 Barley Mow Passage, Chiswick, London W4 4PH.
Tel: (Advertising/Editorial/Production) 081 994 6477
(Subscriptions) 0442 824501
Facsimile: 081 994 1533 Telex: 8811418 SPACES G
ISSN: 0268-6872

Subscriptions

.EXE Magazine is a monthly journal for software developers. It is available only by subscription, at a cost of £35 per annum (11 issues) in the UK, £50 for 2 years. Overseas subscriptions cost £45 per annum, £70 for 2 years. 'A subscription implies that this journal will be sent to the subscriber until one of the three expires' - AG Macdonell. The magazine is published around the 26th of the month preceding the cover date. There is no January issue. To subscribe or obtain details of overseas/academic rates, please call 044282 4501, or write to The Subscriptions Manager, .EXE Magazine, 10 Barley Mow Passage, Chiswick, London W4 4PH. We can invoice your company or take ACCESS and VISA cards. In case of a query about your subscription, please call 044282 4501.

Back issues are available at a cost of £3.50 each - call our Chiswick office (081 994 6477) for a list of issues still in print.

Editorial

Editorial enquiries should be addressed to The Editor, .EXE Magazine, 10 Barley Mow Passage, Chiswick, London W4 4PH. We welcome letters, opinions, suggestions and articles from our readers. Please write for a copy of our Contributors' Guide.

Information contained in .EXE is believed to be correct. If errors are found, we will endeavour to publish a clarification in the next available issue.

From time to time, we offer to copy the PC program code described in an article onto diskette for our readers. In this case, please send a blank, formatted disk with a self-addressed, prepaid mailer to the editorial address given above. We can copy both 5.25" and 3.5" disks.

The publishers can accept no liability for any consequences of using software distributed in this way.

Advertising

If you are interested in advertising in this magazine, please write to the Display Advertising Manager/Recruitment Sales Manager (as appropriate), .EXE Magazine, 10 Barley Mow Passage, Chiswick, London W4 4PH, or call 081 994 6477 for details of our advertising rates.

Pronunciation

.EXE Magazine rhymes with 'not sexy magazine'.

Copyright

Issue theme: Object Oriented Programming

The Reality of OO

Alan Sizemored avises developers to look before they leap onto the object oriented bandwagon if they are to avoid disappointment.

Much interest surrounds the subject of object oriented technology. The bandwagon on which it rides is, however, moving too fast for users to understand where it's going, and if it's worth paying the fare to climb aboard. At blame is the marketing hype which has consistently pushed the market along too fast and too soon. So while few truly know whether object orientation is good for their businesses or if too much can be positively harmful, everyone feels compelled to tow the OO line. Fortunately, the marketing frenzy is at last dying down. As a result, it is becoming possible to take a less subjective view of OO. At the very least, software developers should now be able to tell whether their applications would benefit from this technology, or whether it would be wise to leave OO well alone.

The only way to decide whether or not OO is 'right' for you is to look at the technology removed from hype. Through doing this, it becomes clear that initial expectations of OO have grown more quickly than the technology itself. While this is not a problem unique to OO, it does mean that developers should look twice at what they expect from it, then examine the technology to ensure it is likely to meet such expectations. More likely than not, the two will not correspond exactly. Recognising this imbalance is half the battle; expectations of OO must be tempered with knowledge of what makes it work. So, my first piece of advice to those thinking of adopting OO is to be aware that OO will not fit easily into your existing working methods. In fact, you will have to change them dramatically.

For example, initial design work is likely to take longer than most managers' expectations; a great deal of time must first be spent building a set of base classes before application specific code can even be started. The length of time taken to lay these foundations is often a 'hidden' factor. But in order for their adoption of OO to be successful, it is vital that engineers and managers are aware of the input required at this stage, as time spent here will result in time savings in other areas such as coding and testing.

Engineers should also realise that OO technology requires a conceptual leap of the imagination. For perhaps the first time it becomes essential to think of objects in generic terms. There is a fundamental difference in approach which OO programming requires, and if not recognised causes OO to prove unnecessarily disappointing. Forget the assurances that, if you can program in C++, you can immediately start to write OO programs. This is the lesson Cadre learned when we started the team-work project five years' ago. Then, we were content to start the OO programme simply by training our engineers to use C++. After some time we realised that just because the engineers were using C++, they were not producing object oriented programs. Instead they merely developed traditional C programs in the C++ language.

If OO technology is to stand a chance of success, there is an urgent need for engineers to move away from the widespread, yet simplistic use of language and think generically. Moreover, companies need to be aware that accommodating such a difference in approach is not a task to be underestimated and will certainly involve extensive training.

Our application lent itself to OO because it had some very object-based characteristics; certain classes of object could be applied to particular classes of operation at a very generic level. However, OO is not necessarily appropriate for every application. For instance, there are some basic number-crunching type applications which an OO approach will not help significantly. But even suitable applications will only benefit from OO if the technology is viewed realistically. Then, and only then, will users be able to reap the rewards. For example, OO supplied Cadre with a product architecture that enabled us to support new design artefacts very quickly. Object code has been re-used in different applications where the same objects appear and maintenance costs have been minimised and productivity considerably improved.

There is no doubt that marketing hype, rather than alleviating difficulties serves merely to aggravate them. Problems with OO will not be solved until they are first recognised; and pushing OO as the perfect solution for all software development projects is not the answer. A more likely influence on OO's acceptance as a genuine alternative is the continuing sophistication and maturing of CASE tools. It cannot be denied that such tools are still developing but, to my mind, there is no doubt that with the correct approach and a clear understanding of the technology, OO has the potential to provide the benefits it has promised for so long. EXE

Alan Sizemore became the MD of Cadre Technologies Ltd in February 1991. Cadre develops, markets and supports integrated software development products to manage and automate analysis, design, code generation and reverse engineering. Cadre can be contacted on 0344 300003.

GREY MATTER

Prigg Meadow, Ashburton, Devon TQ13 7DF

Prices do not include VAT or other local taxes, but do include delivery in the UK and Europe.
Please check prices at time of order as ads are prepared some weeks before publication.
This page lists some of our products– call us for a complete price list and details of discount structure.
ORDER BY PHONE WITH YOUR CREDIT CARD.

TEL: (0364) 53499

FAX: (0364) 53071

CROSS ASSEMBLERS

We supply C Cross Compilers by 2500AD, Avocet, Aztec, Lattice, IAR and Hi-Tech hosted on DOS and targeted on Z80/64180, 6502, 6801, 68HC11, 6301, 6809, 7811, 8051, 8096 & 68000/020. Also cross-assemblers by Avocet, 2500AD, Crash Barrier (METAI) and IAR Systems hosted on DOS, CP/M-86 and CP/M-80 with over 30 target processors. In total over 200 products with no space to list them here. We hold some stock but you should allow 10-14 days for delivery. Please call for information or advice.

ADA COMPILERS

Janus/Ada Compiler	DOS	£85
Janus/Ada PDS	386->DOS	£385
OpenAda DOS	DOS	£260

C LIBRARIES & TOOLS

Comms

C Asynch Manager	DOS	£120
C Comms Toolkit	DOS	£89
Essential Comms	DOS	£170
Greenleaf CommLib 3.2	DOS	£195
Silvercomm C Asynch	DOS	£140

Database

Btrieve	OS2,Win,DOS	£345
C-tree Plus	DOS	£340
CodeBase 4.5	DOS	£215
D-ISAM	DOS	£325
Faircom Toolbox Prof	DOS	£680
Pro-C v2.1	DOS	£145
Pro-C v2.5	DOS	£495
Raima Data Manager	Win,DOS	£425
SoftFocus Btree/ISAM	DOS	£80
Sycero.C Network	DOS	£895

Graphics

Essential Graphics Kernel	DOS	£130
Essential Graphics GUI	DOS	£105
GraphiC 6.0	DOS	£305
graphics-MENU & Data Entry	DOS	£180
GSS GDT	OS2,DOS	£425
GX Graphics	DOS	£120
HALO Professional 2.0	DOS	£270
MetaWINDOW	DOS	£165
PCX Toolkit	DOS	£150
Real-Time Graphics & M/CT	DOS	£145
TEGL Windows Toolkit II	DOS	£70

Screen

C Windows Toolkit	DOS	£59
C-Scape + Look&Feel	OS2&DOS	£370
C-Worthy	DOS	£389
Greenleaf Datawindows	DOS	£215
MEWEL Window System	DOS	£195
MEWEL Toolbox	DOS	£165
Object Professional for C++	DOS	£150
Panel Plus II	OS/2&Win&DOS	£260
ProtoGen	Win	£60
ProtoView	Win	£330
TCXL-DOS 6.0	DOS	£75
TCXL Professional 6.0	Win&DOS	£195
Vermont Views V3.0	DOS	£325
Vitamin C V4.0	DOS	£250

General & Systems Libraries

C Tools Plus/6.0	DOS	£99
CodeRunner	DOS	£140
Heap Expander 3.0	DOS	£85
Hold Everything	DOS	£130
PDK £1, £2, £3	DOS	£70
PowerSTOR	DOS	£85
Resident-C	DOS	£135

Math & Scientific

C/Math Toolchest & Grafix	DOS	£45
Huge Virtual Array & NAT	DOS	£145
Mathpak 87	DOS	£85
Science & Eng Tools	DOS	£60

Tools

C-Clearly	DOS	£85
Clear+ for C	DOS	£140
CodeCheck Prof	DOS	£435
LALR Pro 4.2	DOS	£275
MKS LEX & YACC	DOS	£165
PC-Lint	OS2&DOS	£85
PCYACC Prof	DOS	£435
Source Print	DOS	£75
Tree Diagrammer	DOS	£75

C & C++ WINDOWS LIBRARIES

Btrieve	Win	£345
CodeBase 4.5	Win&DOS	£215
dAnalyst Gold	Win&DOS	£295
POET	Win	£299
C++/Views	Win	£270
GUI Master	Win	£380
ObjectGraphics for C++	Win	£159
ProtoGen	Win	£60
ProtoGen for Borland C++	Win	£39
ProtoView	Win	£330
Win++	Win	£160
Windows Control Palette	Win	£99
WindowsMaker Pro	Win	£535
Zinc (inc Source)	Win&DOS	£325

DISK COPYING SERVICE

We can copy files to and from 600 disk formats including CP/M, CP/M-86, MS- DOS, PC-DOS, APPLE, SIRIUS, BBC, TORCH, APRICOT, HP-150, TRSDOS, AMSTRAD, ATARIST, MACINTOSH.

Our charge is £10.00 + disk + VAT with discounts on small quantities and disks are normally despatched within 24hrs of receipt.

For more information call us.

PROGRAMMING TOOLS

Ada
Assemblers
Basic
C++
Comal
Cross Dev
Debuggers
Forth
Graphics
Linkers/Locaters
Logo
Pascal
Rexx
Snobol4
Windows
Algol
AWK
C
Cobol
Comms
Database
Editors
Fortran
Icon
Lisp
Modula-2
Prolog
Smalltalk
Version Control
Yacc & Lex

We stock many items for which there is no space in these advertisements.

DEBUGGING TOOLS

NEW SOFT-ICE/W lets you debug VxD's, drivers and interrupt routines at source level, interactions between DOS TSR's and Windows Apps, programs in DOS boxes and displays valuable system information (including the complex internal structures of Windows). EVERY WINDOWS PROGRAMMER SHOULD HAVE ONE. CALL US NOW.

Bounds Checker	386	£220
Multiscope for DOS	DOS	£140
Multiscope for Windows	Win&DOS	£270
Multiscope for OS/2	OS/2	£315
Periscope I/512K	DOS	£490
Periscope II	DOS	£150
Periscope II-X	DOS	£130
Periscope/EM	386	£195
Quaid Analyser	DOS	£130
Soft-ICE	386	£340
Soft-ICE/W	Win-386	£340
Soft-ICE & Bounds Checker	386	£470

ASSEMBLERS

MS MASM PDS v6	OS2&DOS	£75
PharLap 386IDOS Ext SDK	DOS	£330
PharLap 386IASM/Linkloc	DOS	£830
Optasm	DOS	£125
Turbo Debugger & Tools	DOS	£89

Not all assemblers are supplied with a linker. Check before ordering.

C & C++ COMPILERS

NEW WATCOM C/386 9.0 includes 486 optimisations and generates PenPoint and OS/2 2.0 apps (as well as DOS-386, Windows and AutoCAD ADS). ONLY £485.

NEW MICROSOFT C/C++ 7.0 should ship at the beginning of April. READ THE REVIEWS. Then buy one from us - we are taking advance orders now. ONLY £222 or £99 for the Upgrade.

Aztec C86-c	DOS	£260
Borland C++ 3.0	Win&DOS	£199
Borland C++ & AFX 3.0	Win&DOS	£295
Borland C++ & AFX U/G	Win&DOS	£189
Microsoft C/C++ 7.0	Win&DOS	£222
Microsoft C/C++ Upgrade	Win&DOS	£99
MS QuickC 2.5	DOS	£50
MS QuickC & QuickAsm	DOS	£83
MS QuickC for Windows	Win	£95
Topspeed C Prof	Win&DOS	£165
Topspeed C++ Prof	Win&DOS	£165
Turbo C++	DOS	£50
Turbo C++ & Turbo Vision	DOS	£69
Turbo C++ for Windows	Win	£85
Watcom C 9.0	OS2&Win&DOS	£275
Zortech C++	Win&DOS	£370
Zortech C++ S&E	Win&DOS	£550
High C 386 Local 1.7	PL386	£585
High C 386 Global 2.3	PL386	£650
Intel 386 C Code Builder	Win&386	£399
NDP 386 C	386	£545
Watcom C/386 9.0	Win&386	£485

LOW PRICES FOR MICROSOFT & BORLAND PRODUCTS

Microsoft Basic PDS 7.1	**£222**
Microsoft C/C++ 7.0	**£222**
Microsoft C/C++ Upgrade	**£99**
Microsoft Cobol 4.5	**£435**
Microsoft Fortran 5.1	**£205**
Microsoft Macro Assembler 6.0	**£75**
Microsoft Pascal 4.0	**£139**
Microsoft QuickBASIC 4.5	**£50**
Microsoft QuickC 2.5	**£50**
Microsoft QuickC & Assembler	**£83**
Microsoft QuickC for Windows	**£95**
Microsoft QuickPascal 1.0	**£50**
Microsoft Source Profiler	**£45**
Microsoft Visual Basic	**£95**
Microsoft Windows 3.0	**£70**
Microsoft Windows SDK 3.0	**£225**
MS Multimedia Dev Kit	**£225**
MS MDK & SB Pro Card	**£365**
Borland C++ 3.0	**£199**
Borland C++ & AFX 3.0	**£295**
Borland C++ & AFX Upgrade	**£189**
Borland C++ & Paradox Engine	**£199**
Borland Corporate Prod Pack	**£330**
ProtoGen for Borland C++	**£39**
Turbo C++	**£50**
Turbo C++ & Turbo Vision	**£69**
Turbo C++ for Windows	**£85**
Turbo Debugger & Tools	**£89**
Turbo Pascal 6.0	**£69**
Turbo Pascal Professional 6.0	**£137**
Turbo Pascal for Windows	**£103**
Paradox 3.5	**£340**
Paradox Engine 2.0	**£223**

with Full Technical Support

PROGRAM EDITORS

NEW MULTI-EDIT V6 includes new C-like macro language, real-time DOS output window, faster virtual memory manager, binary & Unix file editing, XMS support, etc, etc. CALL US FOR MORE DETAILS, including site licences, update pricing.

NEW MULTI-EDIT ADD-ONS: MULTI-TAGS hypertext source code browser @ £29, EVOLVE for dBase or Clipper @ £65, GUIDE context-sensitive on-line hypertext reference manual for C++ @ £35 or Turbo Pascal 6 @ £20.

Brief	DOS	£199
EC Editor	DOS	£40
Epsilon v5	DOS	£155
Kedit v4	DOS	£95
Multi-Edit Lite v6	DOS	£49
Multi-Edit Standard v6	DOS	£75
Multi-Edit Professional v6	DOS	£105
Norton Editor v2	DOS	£45
PVCS Prof Editor	DOS	£195
QEdit Advanced	DOS	£45
Vedit-Plus v3.5	DOS	£120

For more information please call us.

CIRCLE NO. 813

News

Client/Server Group
The British Computer Society has set up a new vendor-independent special interest user group which aims to spread the word on Client/Server technology. The Client/Server Group (CSG) will hold conferences and evening meetings and will provide information on emerging standards, vendor strategies and product details. There will also be a quarterly newsletter. For more information, contact Nick Evans, Chairman of the CSG, on 081 9938080.

Motif++
View.h++ and MouseWrapper.h++ are two new C++ class libraries from Rogue Software Inc. View.h++ is an object-oriented implementation of Motif, based on the Model-View-Controller architecture, and provides several built-in dialog boxes to enable the creation of 'instant' file dialogs and lists. MouseWrapper.h++ is a library for building interactive applications. It provides routines for picking, dragging and rubber-banding. The libraries are available for the RS/6000 and Sun Sparcstation. View.h++ costs $795 and MouseWrapper.h++ costs $495 (per seat). Rogue Software can be contacted on 0101 503 7543010.

Arago
US-based WordTech Systems has announced a family of dBASE IV compatible products under the strange name 'Arago'. The components of the system include a dBASE IV interpreter (dBXL), a compiler (QuickSilver - WordTech claims that this is 'the first and only dBASE IV compatible compiler on the market'), an event-driven Panel Painter, the excellent R&R Report Writer and a bundle of other tools. The whole lot, named 'Arago Professional', retails in the US for $1,199, but some components are available separately. The US phone number is 0101 510 254 0900, or fax 0101 510 254 0288.

New Vines Range
Banyan Systems has increased its range of networking software by adding 5, 10 and 20 user versions of the popular Banyan Vines networks. Vines supports local and wide area networks, server-to-server communications and connectivity to minis and mainframes. Banyan Vines includes software which lets networked printers connect to local DOS and OS/2 workstations. Vines costs £970 for the 5 user version, £1870 (10 user version) and £2994 (20 user version). Banyan Systems can contacted on 0293 612284.

Borland bits

Borland has fortified its C++ line-up with the introduction of a low-end Turbo C++ 3.0 for DOS. Main new features are: the compiler's editor offers colour syntax highlighting, AT&T V3.0 language compatibility (ie 'proper' templates, as opposed to the macro equivalents in 2.1 compatible MS C7.0) and pre-compiled headers. RRP is £69.95, but you'll have to make an extra £29.95 for Turbo Vision. There is also an upgrade deal for £39.95.

Borland continues to be very much on the attack regarding Microsoft's C7.0. In an interview with *.EXE*, company spokespersons Gene Wang (Vice President and General Manager of Borland's Languages Business Unit) and Paul Gross (Head of C++) criticised the Microsoft Foundation Classes hierarchy for being 'such a thin layer over the Windows API as to be next to useless... offering no real benefits in terms of code reduction or speed of development'. Mr Wang further stated that Microsoft was not at all committed to OOP. (For a Microsoft angle on Borland, please see our interview with Dave Weil elsewhere in this issue).

As to the company's own plans, Borland said that there will shortly be an intermediate release of Borland C++, V3.01, with some bug fixes and improvements. There are also plans to make some extra class hierarchies available that the company uses internally. These include spreadsheet classes that are used in the development of Quattro Pro.

Hardcopy Library

Obtain instant hardcopy graphical output with AnSoft's PGL printer graphics library. PGL supports colour as well as monochrome output on many devices including LaserJet, InkJet, PaintJet Dot Matrix and PostScript printers. It can also output PCX files and includes a screen preview facility. There is support for both parallel and serial printers and the library provides several routines for controlling the printer. PGL uses virtual device coördinates so that input is independent of the output device. PGL can be used with many development languages including C, BASIC, Pascal, Fortran, Clipper and assembler. The PGL ToolKit costs $195 and is available from Ansoft on 0101 301 4702335.

Micro Probe

Ashling Microsystems has developed the Code Coverage Test Probe, a new device that allows programmers developing code for embedded systems to determine which lines of code in a program have been executed. The probe is attached directly to the microprocessor and communicates with a PC via its serial link. A copy of the software to be tested is loaded onto the PC and, as the software runs, the probe generates a report. The report can be produced in various format and indicates which lines of code have been tested and the duration of the test.

The Code Coverage Test Probe costs IR£499 and is available for the 8051, 68HC11 and 80x96 microprocessors. There are also plans to support the H8, Z80 and 68000 processors. Ashling Microsystems Ltd is based in Ireland and can be contacted on 010353 61 334466.

ObjectGraphics++

Whitewater has launched the C++ version of its ObjectGraphics library (previously available only for Turbo Pascal for Windows - see *.EXE Oct 91* 'Windows Graphics Without Tears'). ObjectGraphics provides an object-oriented replacement for the Windows Graphical Device Interface. There are several built-in shapes and a number of rendering tools like 'Pens', 'Brushes' and 'TextPens' for filling these shapes. ObjectGraphics operates in conjunction with Borland's Object Windows Library and costs £129 (£255 with source code). ObjectGraphics is distributed in the UK by The Software Construction Company on 0763 244114.

Watcom V9.0/386

Canadian-based Watcom has announced V9.0 of its cult optimising 32-bit C compiler. Here's a compiler which has everything except that ++ postfix: a bundled, licence-free version of the (DPMI compatible) Rational DOS extender, OS/2 2.0 compatibility, the ability to produce Windows 3.0 compatible 32-bit applications and DLLs, an inline assembler, a debugger, protected mode linker, 486 instruction sequencing and so on. US price for package is $895 - call UK distributors Grey Matter (0364 53499) or System Science (071 833 1022) for UK price and availability.

Last time we looked at this compiler, although good in other respects, it proved a little disappointing in its performance - but that was two versions ago, since then Watcom has requested a copy of our test code and, we expect, put things to right. We intend to schedule a re-review in the near future. Also: when *are* you going to do a C++, guys?

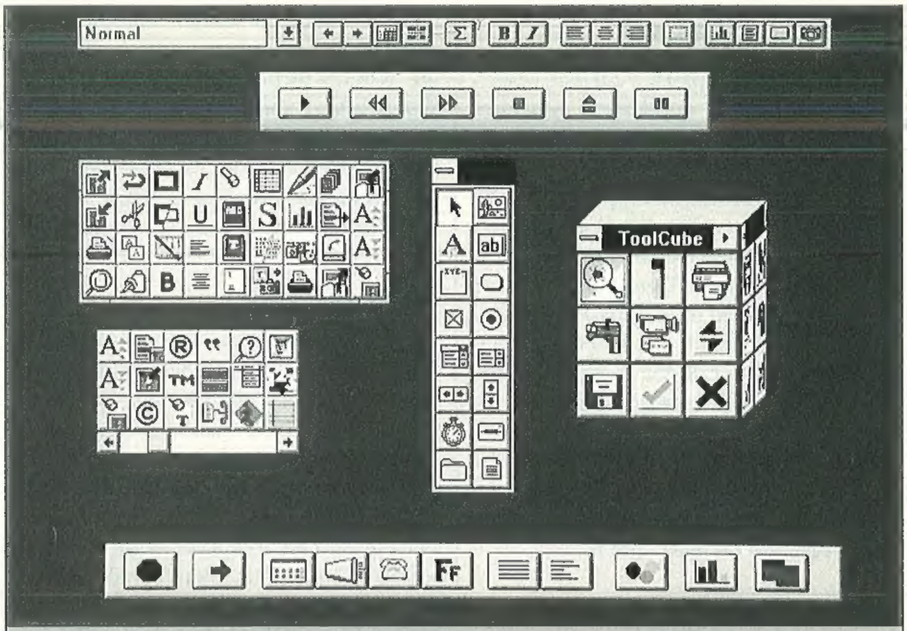
XRAY Source Explorer
Microtech Research, best known for its XRAY range of embedded debuggers, has announced a new tool for C code analysis. The XRAY Source Explorer enables the programmer to browse the C function call hierarchy graphically and examine function definitions pulled out from the source. Source Explorer currently runs on Sun SPARC under both Motif and Open Look, and is priced from £750 for a single-user version. Microtech is on 0256 57551.

Support for ZTC
Sycero C can now generate C code for the Zortech C++ compiler. Using Zortech's Virtual Code Manager, Sycero C should be able to generate much larger programs, capable of running in DOS-protected mode. Sycero C enables developers to create C applications for DOS and SCO UNIX/Xenix. Sycero C for Zortech C++ costs £595 and is distributed by System C Limited on 0622 691616.

Vleermuis GUI
Holland-based Vleermuis Software Research has launched GUI_Master V2.0, the latest incarnation of its GUI class library for building PM and Windows applications. GUI_Master now provides Application Builder, an interactive design tool for creating applications on screen. There is also a C++ class browser. GUI_Builder costs $2500 (£1412). Vleermuis is on 01031 30 324944.

Yacc-o-scope
Yacc tracker is a new debugging tool bundled with the latest version of MKS Lex & Yacc (V3.1), which lets you trace through the parser that Yacc generates. This enables you to step through the parsing of a given input string interactively in order to detect any flaws in your grammar. MKS Lex & Yacc V3.1 for DOS costs £225. Existing users can upgrade for £50. MKS Lex & Yacc is distributed in the UK by The Software Construction Company on 0763 244114.

C Supercomputing
3L, the Scottish manufacturer of parallel software for the Inmos transputer family, has launched a version of its Parallel C compiler for Texas Instrument's C40 processor. With blazing performance claims of around 275 MIPs and 50 MFLOPS, 3L hopes that developers using Parallel C for the INMOS family will migrate to the faster Texas platform with relative ease. Parallel C for the C40 costs £2500. Phone 3L on 0506 415949 for details.



Rubik's Windows

Our picture shows the winning entry for our 'Sexiest Windows Controls of the Month' competition. TbxSHIELD is a toolkit for Windows and, unusually, for OS/2 PM. It contains a set of pre-defined controls including virtual, scrollable toolboxes, tool bars, icon palettes and button bars. We particularly admire the rotating ToolCube, which gives you the opportunity to baffle your punters with thirty-six incomprehensible little pictures where previously you could have only fitted nine.

TbxShield is a DLL, so you should be able to get at it from Microsoft's Visual Basic and Whitewater's Actor as well as lower-level languages. The package is distributed in the UK by Systemstar (0992 500919) and costs £255 for either a Windows or PM version.

Customise VB

Add a personal touch to your Visual Basic applications with the Custom Control Factory (CCF). Controls are created interactively within VB and these can be made to respond to any sequence of input triggers in addition to the Clicked, Double-clicked and Dragged events provided by VB. This enables a control to have several states, each with its own unique appearance. The controls can be created using text, bitmaps and icons. Any control attribute can be changed at run-time and CCF provides facilities for image and automatic control sizing. The Custom Control Factory is priced at £49 and is available from The Software Construction Company on 0763 244114.

C/C++ Tester

Cantanta is a program verification tool for C and C++ which offers three different types of testing. First, Cantata can perform Static Analysis of the source code using a set of criteria which the user must specify. This can be used to highlight areas of code inefficiency or over complication. Next, Dynamic Testing allows the developer to check that the application produces the correct results. Finally the effectiveness of the Dynamic Testing can be assessed with a Coverage Analysis.

Test scripts are written in a subset of C/C++ and testing can take place in a target/host environment to assist in the development of embedded systems. Cantanta costs £7000 and is available on many platforms including DOS and UNIX. For more information contact IPL on 0225 444888.

DOS Windows

Decos Software's Decos/Graphics is a new GUI library for DOS-based machines which is compatible with the XVT GUI libraries. This means that applications developed with Decos/Graphics can be ported to any of the other XVT platforms including Windows, PM, Macintosh and Open Look. It is based on event-driven architecture and supports CGA/EGA/CGA graphics cards. There are 103 fonts, a Resource compiler which uses XVT's Universal Resource Language (URL), and a Help compiler. URL enables resources to be shared between other XVT platforms without re-compilation. Decos/Graphics costs $795 and supports Borland C++, Turbo C and Microsoft C. Decos Software Engineering can be reached on 0101 407 3670407.

Win O-O App Gen

IntelliCorp 's Kappa-PC V2.0 is an object-oriented Windows application generator which produces C source for the Microsoft C V6.0 and Borland C++ V3.0 compilers. The functional blocks in Kappa-PC are provided as DLLs to enable distribution of runtime code independent of the Kappa-PC support environment. Third-party DLLs can also be accessed in the Kappa-PC Application Language (KAL). Kappa-PC provides built-in support for dBASE, Lotus 1-2-3 and Excel files. It also contains a DLL which provides a link to Techgnosis's SQLLink.

Kappa-PC costs £2500 and is due to be released at the end of May, to coincide with the availability of Windows 3.1. IntelliCorp can be reached on 0962 735348.

Windows Futures

Reports of its exact release date vary, but we expect Windows 3.1 to be out by the time you read this. Looking further forward, Microsoft has already published the API for Win32, both in book form and as a giant (2.5 MB) help file, downloadable from bulletin boards (you'll find it in the `windows` forum on CIX, and in the MSOPSYS forum on CompuServe).

Microsoft has also announced another package designed to encourage the take-up of Win32: the Win32S development tool. This will consists of a full 32-bit version of C 7.0, a DLL and a VXD (Windows virtual device driver). Using these, the developer can write 32-bit Windows programs for Windows 3.1 - calls to the Win32 API are 'thunked' to 16 bits at a performance overhead of around 8% (says Microsoft). Functions, such as multithreading, that are not supported by Windows 3.1 are stubbed out to return failure. When Windows NT comes along, Win32S applications should run native at full speed. Microsoft will permit the DLL and VXD to be distributed licence-free with applications.

An exact date for Win32S's release was not given - after all, the 16-bit version of C7.0 has been delayed several times - but it is to be expected sooner. Microsoft indicated that it probably wouldn't be licensing the Win32S technology to other vendors. When asked about this, Borland's Gene Wang said that he thought it would be easy for compiler vendors to implement their own versions, pointing out that Watcom had already done so (see Watcom story).

Baby AS/400

Break away from the AS/400 with Baby/4XX from California Software Products. Baby/4XX provides an AS/400 system on a PC which enables applications to be developed on the PC and later ported to the AS/400. It is also possible to migrate existing software from the AS/400 to the PC. Baby/4XX can integrate with PC-based software and offers client/server facilities when used with LAN Manager. It provides many AS/400 features including the RPG400 compiler, Control Language compiler, Data Description Specification, Programming Development Manager and Database Management support. The Full development system costs £2500. A runtime version is also available at £650. California Software Products is on 0844 274123.

Like minded people

'Please could you put in a news item re: the newly resurrected .EXE Reader Meetings', writes Helena Adams of our Reader Services Dept. Certainly. What would you like us to say? 'The first in a (hopefully) monthly series will be held 5.30pm - 9.30pm of 20th May at the Earls Court Park Inn, London, after System C's "PC Database Day". There will be xBASE-oriented technical speakers, a panel and discussion session, a light supper and a "chance to meet like minded people" (puke! Will - think of something better here - H)'. Seems ok to me. The cost, Helena adds, will be around £35, and everybody will be going down the pub afterwards.

For more info, please ring Helena on 081 994 6477 x2340. To find out about the PC Database Day, call Judi Holly on 0622 691616.

UNIX/DOS Streamer

Object Library is a new terminal emulation tool from Planar Systems which provides a mechanism for transferring data between a UNIX and a DOS application character-by-character. `stdio` from an application running under UNIX is picked-up by Planar's UNIX-Terminal (UT) terminal emulator which re-routes it to a running DOS application. It is also possible to send messages, generated by a DOS program, to a UNIX application which can then take the appropriate action. Planar has suggested that this could form the basis of a Client/Server system. The Object Library allows a UNIX application to access PC resources such as the screen or hard disk directly by sending escape sequences which are interpreted by a TSR on the PC. There are a number of template routines in the Object Library for reading and writing to these TSRs. UNIX-Terminal is priced at £130 per user. A site license for Object Library costs £1495. For more information contact Planar Systems on 0727 810300.

Connecting...JPI

Developers who use the TopSpeed family of compiler products can now receive the latest information and download bug-fixes by dialling into a new Bulletin-board service from JPI. The annual subscription costs £89 and includes a monthly report of known bugs. Limited access to the JPI SmartLine can be obtained by dialling 071 4791801 (9600 or 1200 bps). For more information contact Jensen & Partners International on 0234 267500.

Link & Go

The Prospero DOS Extender Kit provides developers with a way to incorporate a DOS extender into their Prospero Pascal and Prospero Fortran applications. Prospero claims that most applications only need to be re-linked in order to reap the benefits of using a DOS extender. The kit supports DPMI, Phar Lap's 286/DOS Extender, Rational DOS/16M and Ergo OS/286 DOS extenders. It also allows DOS applications to run within Windows by providing a window for standard I/O. The Prospero DOS Extender Kit V2.1 costs £120. Prospero Software is on 081 7418531.

J comes to life

Readers who have been attending properly will recall the J language, an ASCII variant of APL which was featured in our series 'The Third Side' in March '91. The author of that article, Anthony Camacho, has written to tell us that a full J interpreter is now available, running on PC, Macintosh, Atari ST, Archimedes and assorted UNIX boxes. 'An Implementation of J' costs £58 inc P&P, more details from Anthony Camacho on 0727 860130.

Touch Windows

Throw away your ageing rodent. Now you can physically touch controls in Windows using Touch-Base's TWdriver package for touchscreens. TWdriver provides the same input as you would get with mouse events so a touchscreen can be used immediately within Windows by loading the driver. TWdriver costs £115 and an API is supplied for developing touchscreen applications. Touch-base is on 0202 733767.

EzWin32

Run 32-bit text-only applications under Windows using Microway's new EzWin32 library. EzWin32 is supplied with its own NDP-386 compiler for Windows. A C, Pascal and Fortran compiler is available. EzWin32 costs £375 and is produced by Microway (Europe) Ltd on 081 5415466.

Borland C++ training ...

... RICHFORDS Training.
For developers, by developers,
for the real world.

We're in a class of our own.

"Norma?"

"Yes John."

"Will Borland object to my vision of a classless society?"

"Why not ask Richfords?"

"... GOOD IDEA!"

RICHFORDS C++ training for DOS & WINDOWS

CALL - 071-922 8819 for schedule & bookings

All trademarks acknowledged.

CIRCLE NO. 814

Letters

We welcome short letters on any subject that is relevant to software development. Please write to The Editor, .EXE Magazine, 10 Barley Mow Passage, Chiswick, London W4 4PH. Unless your letter is marked 'Not for Publication', it will be considered for inclusion in this section.

Co-ops

Sir,

Mr Papazissimou's letter in your February issue rang a bell - there must be many 'mature' programmers, analysts, technicians etc who are unemployed or 'self-employed' and at a somewhat loose end. May I suggest that those of us who are interested in generating prospects and income should form a 'co-operative' with the aim of sharing programming skills, work and profit.

It would be much appreciated if anyone who is interested in the proposition would drop me a line stating age, areas of expertise, languages of choice and of experience, available development systems and ideas as to what direction we should take.

As a starting point, may I suggest that any prospective 'collaborators' look at possible projects which will appeal to a wide market segment and which will produce a reasonable income - t hat rules out games and shareware astrological charts! Large projects for a single customer could also be attractive.

Before anybody turns up his nose, may I remind you that there are many projects which one or two people simply cannot undertake because of volume of work but which, if split into 20 or 30 segments (and properly coördinated) can be completed very quickly and efficiently and at a much lower cost than by a overhead-burdened large company.

Charles Treen
Treen Technology Ltd
Bernadotte, Gay St Lane
Pulborough, W Sussex RH20 2HL

Sir,

With the current recession there must be many others out there between jobs, like myself, kicking their heels and wishing they could be involved in a useful and creative project. Furthermore, additional work experience would surely look good on a CV if a suitable job did happen to come along.

I have, in the past, been involved in developing and publishing my own software, but I can assure anyone who is tempted to take this route that it is very difficult to get anywhere with it. Not only do you need contact with other people for ideas, but it is almost impossible to do it all by yourself.

Which is where we come in. Most of us probably have PCs with various development tools at home. If we have nothing better to do then there is no risk in committing our time and labour. Even if there is no immediate prospect of any reward, by pooling our skills and resources, we do have the possibility of a reward in the future.

Obviously the rewards are indefinite and nothing like as bankable as a regular pay cheque - but the fact that they are *possible* must be a better prospect than doing the housework yet again.

I would be very keen to hear from anyone who has any of the necessary skills with a view to some sort of collaboration. If you are interested, please send me your CV, together with any ideas you have for suitable products, platforms and/or low-cost marketing strategies, etc and I will send mine back to you or perhaps put you in touch with someone else who might fit you better.

Mike York
Carn House
Canworthy Water
Launceston
Cornwall PL15 8UB
Phone 0566 81511

We wish Messrs Treen and York the best of luck with their schemes, and confidently look forward to reporting huge successes in the near future - Ed.

Editor Hiccup

Sir,

Cliff Saran's review of editors (Feb 92) was good but for the fact that he doesn't read the manuals. I quote, 'SPE provides all the usual cursor controls and editing commands, although it doesn't provide a column copy facility.'

Let me outline a few of the possible methods. First, in Wordstar emulation mode, position the cursor, type ^KN and move cursor to mark block. Second, in Brief emulation mode, position the cursor, type ALT-C and move cursor to mark block. Third, in MS Word emulation mode, position cursor, type SHIFT-F6 and move cursor to mark block etc...

Perhaps before releasing a review it may be appropriate to check its contents with someone who knows the software to avoid embarrassment.

Clive Collie
Readmar Systems

Cliff Saran replies - I apologise for the inaccurate information. In my defence, I should like to point out that the benchmark was carried out using the native SPE keyboard configuration which does not provide a direct method for column copy via the keyboard (columns can be selected using the mouse or by invoking the appropriate macro).

Malaprop corner

Sir,

While I like the reformed presentation of your magazine, and certainly yield to no-one in my respect for Chris Sennitt and all his works *(see 'Towards a fully object-oriented Clipper, .EXE February '92)*, I cannot let slack editing of contributors' use of English go without comment.

Sennitt uses 'enormity' as if it means 'immensity' and 'simplistic' as if it means 'simple'. This isn't just a pedantic quibble: the words he actually used have significantly different (and more powerful) meanings from his intentions. What will he say when he really meets an example of enormity?

Or am I being simplistic?

Conrad Cork
Stamford

I admit the enormity of my error etc etc - Ed.

Cross-DOS problems

Sir

Re: Michael Towles' letter concerning Clipper and DR-DOS. I have experienced cross-DOS version inconsistencies with

Deep C Diving.
Without the lead boots.

When you've been in the Compiler business as long as we have you find out a few things. Like the importance of DOS and Windows Development Environments and DPMI support. That's why with half a million programmers already using Borland C and C++ products, our 3rd generation C/C++ compiler is about to make an even bigger splash than ever.

Borland C++ 3.0
£299.95 + VAT

Borland C++ 3.0
and Application
Frameworks
£439.95 + VAT

SPECIAL PROTOGEN OFFER only £39.95 (+ P&P & VAT) Call 0800 212727 for details

As well as all the features you would expect from a professional compiler, such as customisable Global Optimisation,for both size and speed, AT&T 2.1 C++ and ANSI standard C compilers, you also get our prize winning Turbo Debugger for DOS and Windows, Turbo Profiler and Turbo Assembler.

And just so you won't get cold feet about diving into the world of C++, we've designed Application Frameworks, full of user interface objects that you can bolt right onto your code to create great looking applications under DOS or Windows. We've even included the source code.

And because we know the importance of fast development, the visual ObjectBrowser lets you see the structure of your code at a glance and navigate through classes, functions and variables to make changes without ever leaving the integrated Windows environment. The EasyWin library even lets you re-compile DOS applications to run under Windows without changing a line of code.

Turbo C++ for Windows £119.95 + VAT

Introducing Turbo C++ for Windows, the quickest way to create Windows Applications. With inbuilt tutorials, context sensitive help and ready to use examples, together with the ObjectWindows Library full of bolt on user interface objects now you can create Windows applications fast. For more information or for upgrade details, take a deep breath and **call FREE on 0800 212727** or complete and return the FREEPOST coupon today.

All Borland products are trademarks or registered trademarks of Borland International. Other brand product names are trademarks of their respective holders. Copyright © 1991 Borland International Inc.

Name ______________________________

Position ____________________________

Company ___________________________

Address ____________________________

____

Postcode ________ Tel. ______________

.EXE 4/92

Please return this coupon to:
Borland International (UK) Ltd.
FREEPOST RG1571, TWYFORD, Berkshire RG10 8BR.
Telephone 0800 212727

BORLAND

The Leader in Object-Oriented Programming

Borland International (UK) Ltd
8 Pavilions, Ruscombe Business Park
Twyford, Berkshire RG10 9NN. Telephone 0800 212727

IRCLE NO. 815

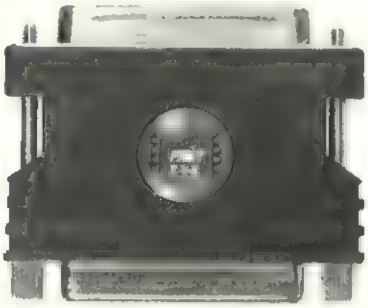
5 out of 5 hackers prefer other software protection methods to Hardlock E-Y-E®

What hackers dislike . . .
Hardlock E-Y-E was designed using cryptographic principles. It took the experience and know-how of Germany's No. 1 in software protection and the leading edge technology of a US semiconductor company to create the ultimate software protection tool. Hardlock E-Y-E is based on a custom chip featuring secure algorithmic response rather than simple bit swapping or counting schemes.

What software developers like . . .
Hardlock E-Y-E combines all the features software developers require in a single product: algorithmic response to provide security and an optional non-volatile memory to allow custom configurations. FAST Electronic has made implementation of Hardlock E-Y-E in your software easy. Use HL-Crypt to protect .EXE or .COM files, or incorporate high level language interface routines in your software. The algorithm parameters and the contents of the memory can be programmed in seconds using our Crypto-Programmer card. This unique card guarantees that no one else can burn your original codes. Simply plug the card into any PC slot and start up your own Hardlock E-Y-E workshop.

What your customers will like . . .
Hardlock E-Y-E allows unlimited backup copies. The device is shipped with the software for the user simply to plug into the parallel interface and forget. Daisy chainability, outstanding reliability (no battery is needed), and the most compact High-Tech design ensure that your customer will accept Hardlock E-Y-E.

What your accountant will like . . .
Hardlock E-Y-E needs no factory coding. This ensures optimum delivery schedules and stock flexibility. Revenues will go up as software piracy and multiple usage are prevented. Despite its wealth of features, Hardlock E-Y-E's prices remain competitive.



Hardlock E-Y-E
programmable, algorithmic response and memory option – all in one.

. . . As more and more software developers, customers and accountants appreciate the Hardlock E-Y-E device, hackers like it less and less.

MAGNIFEYE

Order your demo unit today. Contact Magnifeye,
235-239 Walmer Road, Walmer Studio #6, W11 4EY, Telephone 071 221 8024, Fax 071 792 3449.

CIRCLE NO. 816

Clipper V5.01 code compiled under DR-DOS and shipped to clients running MS-DOS V5.0. The problem disappears when re-compiled under MS-DOS. And no, I didn't stop to work out why, or even bother to call! MS-DOS V5.0 works both ways.

George Walker
Stubbs Systems

Nintendo

Sir,

I have read several complimentary copies of your magazine. I have not subscribed as I feel that both you and your readers are the sort who ask for Nintendo at Christmas. A professional developer is interested only in building systems as efficiently as possible, and as painlessly as possible, not in playing with computers.

Where are the developer friendly application generators previewed? Where are formal design and specification issues addressed? Where is any responsible campaign against the use of specialised lower-level languages for straightforward general business users? Where is the serious issue of multiplying complexity for the sake of cosmetic finish looked at?

Roland Hitchcott
Sussex Consultancy
Bournemouth

An old, old problem

Sir,

I feel I must comment on John Barber's Soapbox *(.EXE, Feb '92)* about the file metaphor. Alternatives do exist. An early interactive file handling system was the Dartmouth Timesharing System, devised at Dartmouth College, New Hampshire in the early 1960s. The word 'file' was not, in fact, used - the model was that you had a 'problem' for the computer, and its solution was a program written in Algol, or in the newly-invented language BASIC. The conversation went something like this (user input in italics):

```
HELLO [wakes the beast]
READY
OLD [gets existing file]
OLD PROBLEM NAME--MYPROG
READY
```

The old program then became your 'current file' and could then be run, altered, replaced, or renamed and saved. Note that the name was associated with the current copy, and if you wanted to change the existing file you 'replaced', whereas to make a new copy you would 'rename' the current copy and 'save' it. The key idea of a 'current copy' of the file has been lost in most systems, but survives in the GE timesharing service which was based on Dartmouth TSS.

This scheme provided some amusement in an early computer scoring project. In 1968, I was sent down to Ruislip Lido for the National Water-Ski Championships. The scoring for jumps was complex, and results were entered into a program which was run from a timesharing terminal in the clubhouse. The lists of competitors were held in two files named MEN and WOMEN, and these were printed out and stuck beside the terminal. The output looked like this:

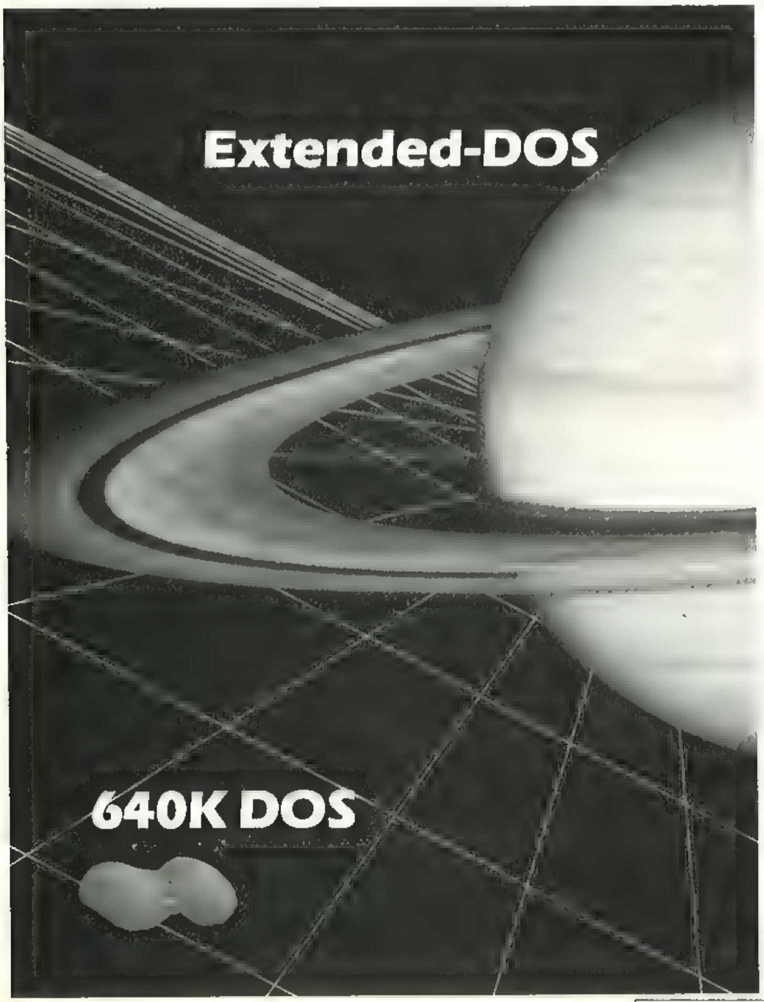
```
OLD
OLD PROBLEM NAME--WOMEN
. . .
```

This occasioned much hilarity among the male officials, but it should be pointed out that there was equally a listing headed: OLD PROBLEM NAME--MEN.

G A Ruscoe
Bedford

Letter of the Month

The writer of the best letter of the month, as judged by the Editor, will receive a £20 book voucher, courtesy of Just Computer Books. The best letter is the one printed first. Please note that letters submitted to this page may be edited.



C7 - The backroom story

At the recent Windows Show, Paul Kemp had the chance to talk about C/C++ 7.0 with Dave Weil, Group Program Manager of Microsoft Systems Languages.

Why has it taken Microsoft so long to bring a C++ compiler to market?

Well, because we did a lot of things, some of which we would do differently with hindsight. We wrote our own C++ compiler from scratch, as opposed to Borland which went out and licensed a front-end. I'm not saying its strategy was wrong. In different circumstances, had we realised just how complicated this was going to be at the start, we might have done it differently. But on the other hand we also cared a lot about having exact control over the language that we support. We wanted it to be a very correct compiler for C++ version 2.1. To be honest, we just did not foresee it as big a development problem as it turned out to be. C++ is not a very well specified language, particularly when you start getting into the nooks and crannies.

Was the removal of OS/2 support a political decision?

No, no; in fact it is not political at all. That is a mistake people like to make because of all the press [coverage] over the IBM-Microsoft situation. We will release an OS/2 1.x hosted version of C7. All we have to do is rerelease the compiler, PWB and some of the utilities. They'll run on OS/2 1.x for people who want to do cross development. Only a small percentage of our users were using OS/2 1.x and a much smaller percentage of those were actually targeting OS/2 1.x applications. Many of them were just using it as a cross development environment because, if the truth be told, C6 was far better running on OS/2 than it was on DOS. We've tried very hard to correct that with C7, so we don't think that OS/2 as a host is quite as critical anymore.

On the other hand, OS/2 is a good development environment. We all know that because we used it for many years internally. So there are reasons why people might want to stay there as a development host, so we'll support that. Consequently we will be releasing an OS/2-*hosted* version. We're not going to *target* it with C7 because we really don't feel there's much new development starting up targeting OS/2 1.x. We will keep C6 available for people who want to target OS/2 specifically.

As far as OS/2 2.0 goes, the answer is, strategically, for us Windows and Windows NT are the key things. OS/2 2.0 becomes a secondary issue. If it becomes a very popular platform and is a big success, I would say we'll probably support it. NT is where we really want people to go, so we're going to support that first.

I have heard that you had some radical rethinks to arrive at the MFC class hierarchy that will eventually ship.

There's quite a story there - what the MFC developers like to call their 'OOP-aholic story'. Basically they were given a mission - they gathered all these hotshot object-oriented programmers at Microsoft and gave them a mission to go off and solve the problems of Windows development with class libraries. So they went off and they researched, and studied, and wrote papers and went crazy. Eventually they figured out all the problems of Windows development and designed a set of class libraries that *they* thought would solve the world's problems.

Then they took it to the Excel group and said 'Hey, we wanna talk to you about how we can help you solve your problems'. They said 'Great, we've got lots of problems, we'd love to have them solved, tell us what's it about?' The MFC guys replied 'Well, you need to do three things; first you need to throw away all your old code; second you know everything you've learned about Windows over the years? - Well forget it; lastly, you know those big machines you've got? - They need to get bigger'.

At the end of all this, the Excel guys just threw them out the door and said 'I think we'll pass'. So these guys went back and said 'Obviously we didn't make ourselves clear, they just don't understand'. So they sat down and said 'We're going to write a bunch of applications'. At the end of that month they all got back together and said 'Boy, that was really hard, I think they were right, this is too difficult'. These guys wrote it and they couldn't understand it all! So they realised they had just gotten a little too enamoured of the powers of object-oriented programming and they'd written a bad set of libraries. So they went back and redesigned their libraries around the existing API, that's now MFC.

What about the message passing mechanism?

We looked at Borland's method and it's fairly efficient, and for a while we considered strongly doing that. But we wanted to stay away from making it implementation-dependant. The problem that I see with OWL is that it requires a Borland compiler, whereas the MFC libraries just require a 2.1-compliant compiler.

Is the Windows C API going to around for some time to come?

Well yes, I think so. At some point I believe it's very possible there will be some C++ APIs, but I think it's a little ways off. The other thing required is that there's got to be a common object mapping for all compilers. That's the main reason we've pushed our object mapping to other vendors very strongly - without a common object mapping you have serious problems with things like DLLs and third party libraries. If I write an executable with Microsoft C++ and a DLL with Borland, you can't call between the two without a common object mapping. Third party library vendors suddenly have to have the Borland version *and* the Microsoft version *and* the X version *and* the Y version and they don't want to do that. That's a nightmare for them in terms of support, stocking and shipping - they'd like to have one library.

We think that's very important and we work very hard with other vendors to make sure that happens. As hard a rivals as we are, even Borland has come around to some extent, they didn't go all the way with

BC++3.0, we'd like to see them do that but we'll have to see

I agree, I think it's very important to get this standardisation.

It was a very hard decision for us to go public with our object mapping because we spent a huge amount of time and developer effort to make it very, very efficient. Had we kept it to ourselves, we think it would have been a significant competitive edge. On the other hand, it wasn't the right thing to do as far as meeting the promise of C++. In the end that won. We also felt that the competition had good ideas that we could use and wanted feedback on. I think Walter Bright at Zortech is putting it in as soon as he has time. We've talked a lot with the people from Watcom, and think that they're going to adopt it but we don't know exactly when.

Unlike Borland, Microsoft seems to be slightly reticent about C++ and OOP.

I like C++ but, frankly, it's not for everybody. I think your average programmer will be fairly intimidated by C++. It's a much more complex language and things happen behind your back. C is an easy language to understand. I can look at it on paper and I can translate it into machine code for you - you can't do that with C++. If, when debugging your application, you want to step through the assembler code you have to have committed the object mapping to memory to have a clue what's going on - it's very, very difficult. I don't think C++ is an ideal language for every application. I'm a very strong believer that different languages are appropriate for different applications. C isn't going to go away.

Microsoft has been drawing a lot of comparisons between C7 and Borland's compiler. Is this a sign that you're worried about market share?

I think that Borland has definitely grabbed the headlines with C++. They make a big deal of 'Objects will cure the world' which I don't frankly believe, but it makes a great story. I think they are a great company. They write very innovative products, they do it very quickly, although they have a great facility for switching stances sometimes, seemingly without making a ripple, so I have to admire them. On the other hand, I've sat through more than a number of Borland presentations where it gets pretty rough if you're a Microsoft employee. So we decided that, okay, if that's the way the game is played, we can play it that way too. But we don't get quite so down and dirty. We're second in the market frankly, so we feel we have to state our case fairly forcefully because they have a leadership position. Given that, we need to make it clear to people that we think we have better technology in many cases and provide a lot of things that they're not providing right now. Yes, we compare them directly to make that very clear.

It must be quite a strange relationship with Borland. Presumably Microsoft, as a company, benefits from having as many Windows development tools out there as possible.

Yes. The Open Tools conference was not something we as a group were very fond of, but we understood the necessity and we participated in it. That's where we first put out our object mapping, making the CodeView formats available publicly and so on.

Was there some arm twisting?

Well, I wouldn't say we were dragged kicking and screaming, but it was clear to us that, potentially, this would cost us a competitive edge. We hated having to give out the object mapping so early, so that somebody actually beat us to market with parts of our own design, but then again, I could say that it's our own fault for not being out there as quickly as we should have been.

EXE

Many thanks for Dave Weil for making some time during his busy schedule to talk to us. Since the interview was conducted, the release of Microsoft C/C++ 7.0 has been postponed once more, and at time of writing is now due in April, with Windows 3.1.

Object-oriented Software Engineering

We have the hype, we have the languages - but do we have a proper development method? Darrel Ince offers a thorough approach to developing OO software.

The last five years has seen a huge amount of progress in object-oriented programming, a technology which promises large savings in the resources used on software projects. However, one problem which still remains is that, although there are now several mature, industry-oriented object-oriented programming languages, there is still a shortage of methods, tools and notations which enable a software developer specify, design and validate object-oriented systems. The aim of this article is to describe a development method.

Before describing the method, it is worth establishing a baseline by reviewing what an object-oriented programming language offers the developer. The main concept that distinguishes an object-oriented programming language from conventional languages such as Pascal and FORTRAN is the class. An example of a class is shown below:

```
class queue
private:
  queuearr:
    array[1..20] of messages;
  queuepointer:=1;
public:
  additemq (m:messages);
  integer countq();
  remitemq (var m:messages);
end class
```

This class implements a queue of messages, say messages in a telecommunication system. The class contains two parts: a private part and a public part. The private part defines items which a programmer is not allowed to access directly. In the class above there are two items: `queuearr`, which holds a queue of messages with the first message held at `queuearr[1]`, and `queuepointer` which indexes the next position in the queue at which a message is to be inserted; this is initially set at 1. The class definition that I have described belongs to no particular language; instead I have taken the liberty of using features found in a number of OOPLs.

The public part of the class describes the facilities which are available to the programmer. First, a subroutine `additemq` which processes a message is declared, this adds the message m to the queue; next, the integer subroutine `countq` is declared ; this returns the number of messages in the queue; and, finally, the subroutine `remitemq` is declared. When this subroutine is called, the head of the queue is placed in the parameter m. The program code for these subroutines is declared elsewhere.

Advantages

Object-oriented programming languages such as C++, Objective C and Eiffel offer a number of advantages over conventional programming languages. First, because it does not allow the programmer to access stored data, an OO system does not contain large numbers of data references which have to be changed whenever, say, a maintenance change has to take place.

Second, the property known as *inheritance* describes the fact that a class can be easily constructed by inheriting many of the properties of a new class. For example, a developer may have a requirement for a class which describes a delivery note for chemical products, but only has in the class library a class which represents a delivery note for electrical goods. It is the work of a moment to change the class in the library to become the required class containing any extra information such as special handling instructions.

Once a software developer has developed a few applications using an object-oriented programming language then, normally, enough classes would have been defined to make up a class library. This library can then be used time and time again in future applications without very much new program code being developed.

Identification

Given that an object-oriented system will contain classes, the major problem is in identifying the objects described by a class, specifying them and designing them. This is the theme of the rest of this article. The first stage is to identify the major entities that make up a system. For example, in an air-traffic control system, typical entities include planes and radars. Once these entities have been discovered, a diagram known as an entity life-history is drawn for each of the entities. This diagram shows the actions that the entity suffers during its life.

An example of this diagram is shown in Figure 1. This shows the ordering of actions on the entity *car*, which is an entity in a system for a car hire company. The top level

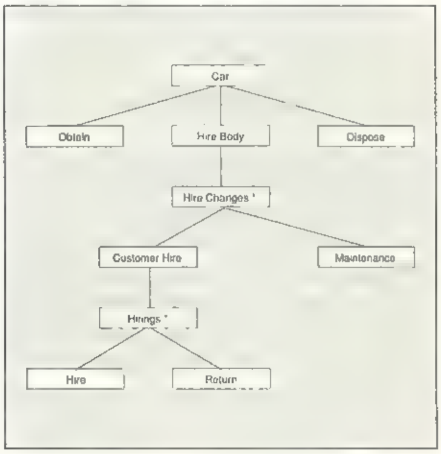
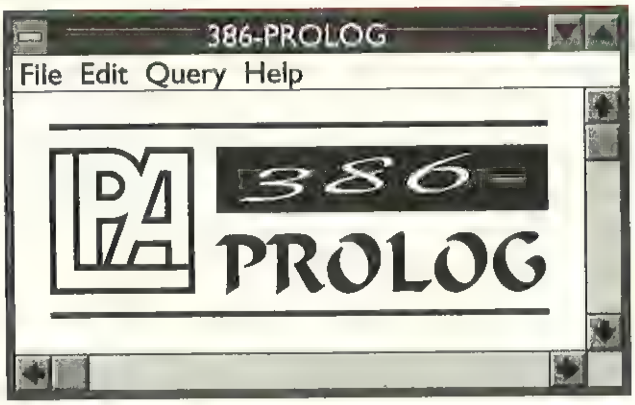


Figure 1 - An example Entity life-history diagram

Windows 3!



Yes, it's here! LPA 386-PROLOG is now available in a version for Windows! Just like the DOS-extender version, it is a genuine 32-bit Prolog compiler which can directly access up to 4G (4096M) of memory. Only this time, it is fully integrated with the world's most popular GUI: Windows 3!

Logic Programming Associates Ltd
Studio 4, Royal Victoria Patriotic Building
Trinity Road, London, SW18 3SX, England
Tel: 081 871 2016 - Fax: 081 874 0449

CIRCLE NO. 819

INNOVATIVE DESIGN TOOLS

FROM GREAT WESTERN INSTRUMENTS

Great Western Instruments are the specialists in innovative design tools for the real-time systems developer. Our services include full technical and design consultancy support for all our products.

THE PROFESSIONAL SUPPLIER

We know as product developers that that real-time software designers expect quality support from their suppliers. We supply some of the finest design tools and language products available and, unlike many suppliers are prepared to back them with full technical support. Join our existing very satisfied customers who regularly benefit from our expertise.

PRODUCTS AND SERVICES

❏ **SELECT** Software low cost **CASE** tools for Yourdon, HOOD and SSADM
❏ **SMS** software management system for MSDOS, OS/2, UNIX and VMS
❏ **Embedded DOS**, the MSDOS v3.31 compatible operating system with multi-tasking
❏ **AMX** Executives for the 80x86, 80386, 68000 and Z80 from **KADAK**
❏ Memory management tools to support EMS, XMS and virtual memory
❏ The **Annabooks** range of BIOS kits, PROMKIT and customised keyboard software

BORLAND Authorised Dealer with language expertise
Authorised **BORLAND** language UPGRADE centre

AUTHORISED UPGRADE CENTRE

GREAT WESTERN
-Understanding the needs of other designers!

For **immediate** information about these & other products in our range, use your Fax/Telephone to call **Facts-direct** (0865) 727232, **Item 444401**

Unit 1m Farrington Fields
Farrington Gurney
Bristol BS18 5UU
Tel 0761 452116
Fax 0761 453226

VAT Extra Carriage Free

All trademarks acknowledged

CIRCLE NO. 820

Blast! The 640K DOS barrier . . .

Let's face it. Today's average PC sports at least 1 meg of RAM, often more. So its natural that your programs should be able to make the most of the many benefits of Extended Memory. Bring on the dynamite – the TopSpeed DOS Extender Toolkit. Not only does it give you an extended development environment, it also gives you a powerful toolkit to produce DOS programs that can make full use of extended memory.

- Provides 512 Mb virtual address space
- Full range of DOS calls
- OS/2 family API
- Supports XMS, DPMI, and VCPI
- Comprehensive porting guide
- Multi-threading and DLL support
- Supports C++, C, Pascal and Modula-2
- Royalty free distribution

For a detailed information pack on the TopSpeed DOS Extender and other TopSpeed languages call us now on 0234 267500.

Much existing code can be ported to the TopSpeed DOS Extended Environment without re-coding. The TopSpeed DOS Extender Toolkit supports TopSpeed C++, C, Modula-2 and Pascal and requires a 286 processor or better.

TopSpeed products are available in the UK from Grey Matter 0364-53499, System Science 071-833-1022, The Software Construction Company 0763-244114 and other leading software tool vendors.

**JPI, The Mansards, Tavistock Street,
Bedford, MK40 2RX
Tel: 0234-267500. Fax: 0234-217094**

* Special pricing for existing users

Blast the 640K DOS barrier with TopSpeed

CIRCLE NO. 821

of the diagram gives the name of the entity. The next level states that a car will first be obtained, will then undergo hirings and finally will be disposed. The box marked *hire body* consists of a series of *hire changes* - the asterisk in the latter box indicates that all the boxes underneath it are to be repeated. The box *hire changes* describes the hiring part of the system: customer hirings followed by maintenance on a car. The box marked *customer hire* describes the fact that a repeated number of hirings is to take place, the asterisk in the box *hirings* again indicates repetition. Finally, the boxes underneath *hirings* show that a customer hire is always followed by a return.

Such diagrams are a good aid to validation and verification: they enable a systems analyst to discover whether any events in the system have been missed. Indeed, in the entity life-history above, an event has been missed: that of a customer crashing a car, and the car being written off.

Annotation

The final process in the construction of an entity life-history diagram is that of annotating it with basic operations. A basic operation corresponds to some operation on an entity, eg updating the maintenance record of a car or some top-level processing code, eg program code which prompts the clerk who uses the car hire system for car details. An example of an entity life history annotated with basic operations is shown in Figure 2. Each basic operation is represented as a square box with a number in it.

A number of these basic operations correspond to operations on a car entity, for example, operation 1 creates a car entity, operation 3 updates the car entity with its make, and operation 7 adds the details of the current customer who has hired the car. Other operations, eg operations 11 and 12, are all concerned with top-level processing.

Abstraction

The operations associated with an entity will eventually end up as public facilities implemented as subroutines. However, before implementation or programming takes place, the entities have to be designed. In the development method described here, they are designed as *abstract data types.* The term abstract data type looks forbidding, but the concept is quite simple.

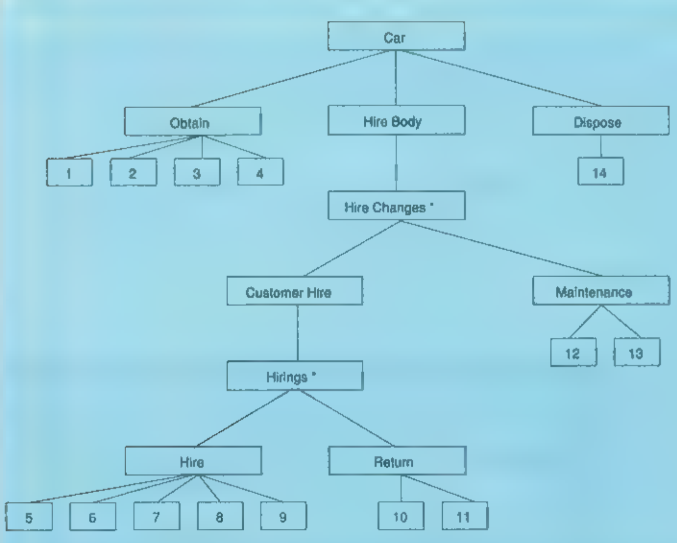
An abstract data type is one which is not normally directly implemented in a programming language. In conventional programming languages, the normal built-in data types are typically integers, reals and characters, together with composite data types made up from these types, for example, arrays of integers, records, and so on. An example of an abstract data type definition is shown in Figure 3. It describes a type which is a queue of spool files used in a multi-user operating system. A spool file is a file of data which is to be printed out on a slow output peripheral such as a laser printer.

The first line states that the abstract data type queue will be a sequence of spool files, the assumption being that spool files have been defined elsewhere. The keywords SEQ OF signify that a sequence of spool files that has some implicit ordering. The next line defines a data invariant: this is a condition that must be true for queues *throughout* the execution of the system that uses queues, namely that after each operation has been applied to a queue, it will remain ordered on some priority criterion.

Following the data invariant is the specification of three operations. The first defines the operation of adding an item to a queue, the item to be added is *item*, the queue to which it is added is *q* (which acts very much like a global variable). The keyword WR indicates that *q* will be written to, ie updated. The text following the definition of *q* defines the processing associated with the operation. The text for the *add* operation describes a condition that must be true *before* the operation is successfully executed and what is true after the operation is executed. The specification of the operation *remove* follows the same pattern.

The final operation is *count*, which counts the number of items in a queue *q*. This operation differs slightly from the others. First, the keyword RD is used, this signifies that *q* will only be read and not updated. The second difference is that there is no text that states what must be true before the operation is executed. The reason for this is that the operation will always be able to execute, irrespective of the number of items in *q*.

The definition of the abstract data type is constructed using an annotated entity life-history. The data part of the abstract data type represents the entity, while the operations in the abstract data type are equivalent to the basic operations in the entity life-history of the entity. After the abstract



Annotations

1. A hire car is created.
2. The clerk at the hire company gives details of the car, eg its registration number and its make.
3. The car is updated with its make.
4. The car is updated with its registration number.
5. The clerk at the hire company gives the details of the customer, eg name, address, telephone number, and so on.
6. The clerk at the hire company gives the hire period of the car.
7. The customer details are added to the car.
8. The hire period of the current hire is added to the car.
9. The car is marked as being hired.
10. The car is marked as having been returned.
11. An invoice is prepared by the system.
12. The clerk at the hire company gives the details of the maintenance that is to take place.
13. The car is updated with details of the maintenance.
14. The car is deleted from the system.

Figure 2 - Annotated Entity life-history diagram

F77L-EM/32 & Lahey Ergo OS/386

Port mainframe programs as large as 96MB to 386/486's with this 32-bit DOS-Extender compiler. The winner of PC Magazine's 1988 Technical Excellence Award just got better. New Version 4.0 includes: Programming Tools, Popular Fortran 90 features, Virtual Memory Support, DESQView support, New Documentation and Free Unlimited Runtime Licenses. F77L-EM/32 and OS/386. *£875.00 plus VAT*

F77L

The fastest real-mode compiler available. F77L can take advantage of your 386 PC by generating 32-bit instructions. New Version 5.0 includes: Fortran 90 features, Weitek support, and Video Graphics. *£375.00 plus VAT*

Lahey Personal Fortran 77

Version 3.0: Full ANSI 77, Editor, Debugger, Linker, Library Manager, Microsoft and Borland C interfaces. A great learning tool at an unbeatable price. *£79.00 plus VAT.*

When people talk about FORTRAN the name mentioned most often is
Lahey

System Science

3-5 Cynthia Street, London N1 9JF
Tel: (071) 833 1022 Fax: (081) 837 6411

CIRCLE NO. 822

CopyControl

THE NEW GENERATION OF COPY PROTECTION

- NO User hassle
- NO Back-up problems
- NO Hardware add-ons or special disks needed
- NO Changes to source code required
- YES CopyControl beats ALL bit-copier Programs
- YES Floppy disks, hard disks and networks supported
- YES CopyControl is totally transparent to the user
- YES You can limit program use by no. of copies, no. of executions or date
- YES CopyControl works on all IBM compatibles
- YES Free demo disk available

For Further Information Phone or Write to

microcosm

Microcosm Limited,
17 Cranbrook Road, Bristol BS6 7BL
Telephone: 0272-441230 Fax: 0272-427295

CIRCLE NO. 823

TopSpeed® C++ TopSpeed® Modula-2 TopSpeed® Pascal TopSpeed® C

"Enough to make you a LATE DEVELOPER"... PC Business World July 91

Imagine a powerful integrated development environment common to whichever languages you choose, capable of supporting DOS, Windows or OS/2 development embodying the latest in OOP technology – it's called ***TopSpeed*** from JPI. Its modular architecture opens up new realms of choice – simply slot-in languages, source libraries, and toolkits to build a development system that meets your needs. No redundant components – what you want is what you get!

***TopSpeed* Environment** – Multi-window editor, powerful Project system, Hypertext help (environment, all languages and library), debugger, syntax checker, menu or command line driven, EMS support, pop-up calc, plus . . . plus . . . DOS or OS/2 **£59**

***TopSpeed* TechKit** – For power programming. Supports DOS DLLs, post mortem debugging, advanced overlay manager, TopSpeed assembler, .EXE file compressor, .OBJ file disassembler. **Windows 3 resource compiler**, Windows 3 run-time libraries, plus . . . plus . . . **£59**

***TopSpeed* C++** – makes C++ lean and mean Unique to JPI – TopSpeed is the only true AT&T 2.1 C++. TopSpeed C++ gives you SmartMethod® Linking which eliminates unreferenced classes, methods and even virtual methods – a real breakthrough for OOP programming. Includes short based pointers, concurrent tasking – even with DOS, plus . . . plus . . . DOS or OS/2 **£59**

***TopSpeed* C** – *the standard is enhanced.* The only ANSI certified C. Generates compact high quality code for DOS, OS/ 2 and Windows 3. Includes run-time error checking, multi-threading, mixed memory models, links to C++, Modula-2 and Pascal, plus . . . DOS or OS/2 **£59**

***TopSpeed* Modula-2** – the world leading implementation of Modula-2. This strongly typed and highly structured language includes type safe conversion between objects, OOP extensions with true multiple inheritance, virtual pointers, based pointers, links to C, C++ and Pascal, plus . . . DOS or OS/2 **£59**

***TopSpeed* Pascal** – the next generation. Power-up your Pascal, convert to TopSpeed, then tune it up with the hottest compiler for DOS, Extended Dos, OS/2, or Windows. ISO 7185 conformant, Turbo to TopSpeed converter, ISO conformant arrays, dynamic strings, separate compilation units, OOP extensions plus much more . . . DOS or OS/2 **£59**

Library Source Kits available in all languages

Call JPI on (0234) 267500 now for your free copy of the TopSpeed 1992 Compiler Catalogue

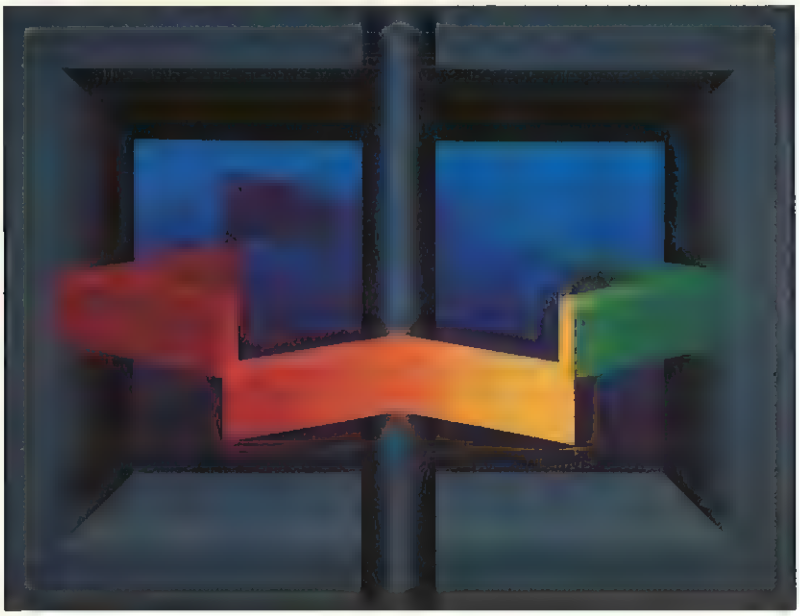
TopSpeed products are available from:
GreyMatter 0364-53499 · System Science 071-833-1022
RTA 081-656-7333

JPI

3 The Mansards, Tavistock Street, Bedford MK40 2RX
Fax: (0234) 217094

CIRCLE NO. 824

Who put C++ to Work?



glockenspiel
class constructors

Glockenspiel C++

Glockenspiel C++ leads the field in object-oriented programming. It's the most effective and efficient implementation around. It conforms exactly to the AT&T C++ specification – and they don't come any closer than that! Portable too: it's on more platforms than any other C++ implementation.

Glockenspiel CommonView®

Glockenspiel CommonView is the leading C++ class library for developing Windows, PM and OSF/Motif applications. Logical, well structured and easy to use, CommonView is 3 to 5 times more productive than the API.

C++ Training & Consultancy

Glockenspiel C++ training courses reflect our experience as C++ developers. From design to development, *learn from the best!* Our C++ consultancy services match your needs as you commit more resources to C++ and object-oriented development.

Further Information

GLOCKENSPIEL, 39 Lower Dominick Street, Dublin 1, Ireland. +353 (1)·733166. Fax +353 (1) 733034

■ **US: IMAGESOFT**, (516) 767-2233. Fax (516) 767-9067. **OASYS,** (617) 862-2002. Fax (617) 863-2633 ■ **UK: QA TRAINING LTD.**, (0285) 655888. Fax (0285) 650537.
■ **Italy INFERENTIA,** (02) 26680568 Fax (02) 2364258 ■ **Sweden· LINSOFT**, (01) 3124780 Fax (01) 3152429. ■ **Germany, Switzerland, Austria· PSI,** (06021) 492-0. Fax (06021) 492-112
■ **Benelux· RIJNHAAVE**, +31 (71) 218121. Fax +31 (71) 216118

Glockenspiel CommonView and Glockenspiel C++ are registered trademarks of Glockenspiel Ltd. The trademarks of their respective corporations are acknowledged.
Glockenspiel "Colour Rotation" logo by Francis Tansey

CIRCLE NO. 825

queue = SEQ OF spool_files
INVARIANT a queue is in ascending
order of *spool file* priority

OPERATION add(item:spool_files)
WR q: queue
For this operation to be defined, *item* must not
already be in *q*. After the
operation has completed, *item* will be
added to *q* in a position determined by its
priority.

OPERATION remove(item:spool_files)
WR q: queue
For this operation to be defined, *q* must
contain at least one item. After the
operation has been completed, the head
of *q* will have been removed and placed in *item*.

OPERATION count(number: Integer)
RD q: queue
When this operation has been completed,
number will contain the number of items in the
queue *q*.

Figure 3 - An example abstract data type operation

data type has been defined, it can be implemented as a class. The abstract data type above would be implemented using some data structure to represent the queue and three subroutines to implement the operations.

Data flow

After all the classes have been written, all that is required is to implement the program code which uses them. This is normally code that implements input/output and switches between the various subroutines that make up a class. A convenient notation to define this processing is a data flow diagram. This is a notation extensively used in many software development methods.

Figure 4 shows an example data flow diagram. It shows the flow of data through a system, and how the data in a system is transformed to other data. My example shows the specification of part of an airline booking system. Users interact with the system by means of a command and a series of parameters. Square boxes represent either sources or consumers of data. Bubbles represent processes that transform data into other data.

First, bubble 7.1 splits a command typed in by a reservation clerk into its components: a command name and a number of parameters. The command name is then checked for correctness: if it is incorrect then an error message is produced; otherwise it is passed on to the part of the system that executes the command.

The parameters are checked by bubble 7.2: if they are correct they, like the command, are passed on for processing. However, if they are incorrect an error is produced that is displayed on a VDU. Do not worry about the numbering in the diagram, all that you should realise is that a data flow diagram is capable of describing what a system should do. Many of the bubbles in such a data flow diagram represent operations implemented as class subroutines. If you have carried out object-oriented development properly, the remaining bubbles are small in number and represent relatively small chunks of program code which are easy to implement.

Life Cycle

One way in which a software developer can sequence the activities described in this article is shown in Figure 5.

First, the statement of requirements for a system is received by the developer. This will describe, in very rough terms, what the system should do. The next stage is to carry out the entity analysis resulting in the development of the entity life-history and basic operations. Next, the entities should be designed in terms of abstract data types. The classes corresponding to these abstract data types are then programmed and integrated. The integration, which can be carried out in parallel with the implementation of classes, will consist of tests that ensure that all the operations in a particular class work correctly with each other. For example, the developer would test that when an item is added to a queue, and then removed, the queue remains unchanged.

After all the classes have been implemented and integrated, the next stage is to write down what the eventual system is to do. The notation used for this is the data flow diagram described above. Finally, the top-level code of the system is designed and constructed. In parallel with these activities, the developer will produce the system tests: these check that the system has been correctly implemented according to the customer's wishes. These tests are then carried out and may be followed by acceptance tests designed by the customer.

This, then, is a typical object-oriented development life-cycle. If you are a single developer or a hacker who enjoys programming for its own sake then you will not need it. However, if you are group of commercial developers or a single programmer who wishes to make money out of your skills, then it should ensure that clean code based on classes is developed; and that, eventually, you will have produced a large number of classes which can be stored in a library and reused time and time again.

Darrel Ince is a Professor of Computer Science at the Open University. Further information describing the techniques described in this article can be found in Object-oriented Software Engineering with C++, *written by Darrel Ince and published by McGraw Hill.*

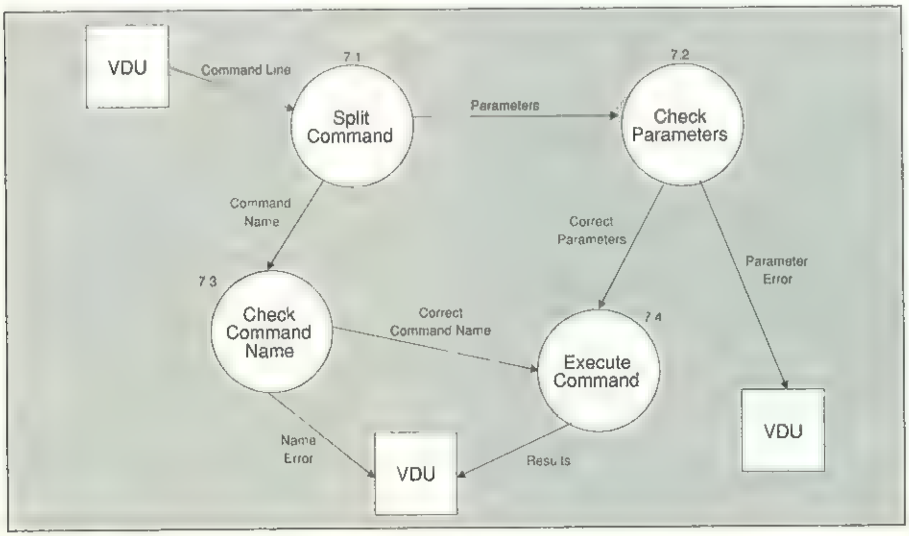


Figure 4 - A data flow diagram

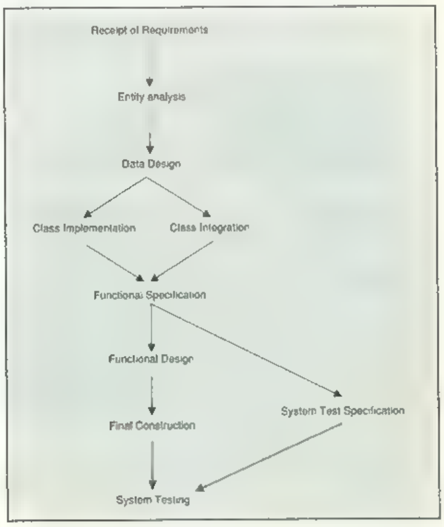


Figure 5 - Development sequence

Turbo Vision - A Supplement

Blaise Computing's Turbo Vision Development Toolkit is an extension to Borland's text mode class library. Willie Watts has been playing around.

Borland introduced the Turbo Vision application framework with Turbo Pascal 6.0 and has latterly ported it (he adds hastily, before all the C++ users switch off) to C++. Turbo Vision (TV) is the definitive OOP text-mode interface library for MS-DOS. You get a complete set of interface objects - windows, scroll bars, menus, radio buttons, dialogs etc - all neatly built around an event-driven architecture. Once sussed (the sussing does take a day or two) Turbo Vision lets you turn out Windows-type programs which suffer from important disadvantages such as being easier to produce, running loads faster and not needing the Windows kernel. You will have deduced that I am sweet on Turbo Vision.

But there are things missing. Fans of Turbo Power's *Object Professional* library sometimes sneer at the absence of specialised input classes, such as date input, although these are easily added by the Turbo Vision user. There is nothing to let you produce a TSR (although the craze for TSRing everything in sight seems to have faded, and the bulkiness of TV code would probably force one to use complex paging schemes). Most of all there's a problem with resources.

Turbo Vision is the definitive OOP text-mode interface library for MS-DOS

TV Resources

Turbo Vision includes a mechanism whereby objects may be made to *persist*, ie they may be saved to disk and subsequently reloaded. This sounds much more simple than it is. Object data contains pointers to virtual method tables and to other objects. These pointers must be fixed up when an object is loaded from disk. To allow this to happen, TV includes base class methods for loading and storing objects, plus a system of object registration whereby each persistent class is assigned a unique integer ID code. TV resources are data files containing numbers of (mixed type) objects, with the objects indexed by name. So the Pascal fragment

```
var
  DlgRes   : PDialog;
  RFile    : TResourceFile;

begin
  {Open Resource file}
  RFile.Init(New(PResStream,
    Init('DRES.BRS',
      stOpenRead, 1024)));
  {Get dialog box}
  DlgRes := PDialog(
       RFile.Get('DLG1'));
  DeskTop^.ExecView(DlgRes);
end;
```

opens a resource file DRES.BRS, extracts the dialog named DLG1 and finally executes it.

Now this is terrific. Dialog DLG1 could contain any number of controls - list boxes, buttons, text input fields and so on - yet the application 'knows' nothing about any of them. We are half-way to writing a purely data-driven program. Also, the standard initialisation code for the dialog, which would be a mess (as it would contain many hardwired coördinates for the size and position of the control components), has been removed from the app prog. The problem is: where has it gone? Using Turbo Vision as supplied by Borland, the programmer must write a separate support program to generate the resource file. All those coördinates must be calculated by hand. Why, the TV programmer might reasonably ask himself, can't I have one of those screen painter programs, as supplied with 4GLs?

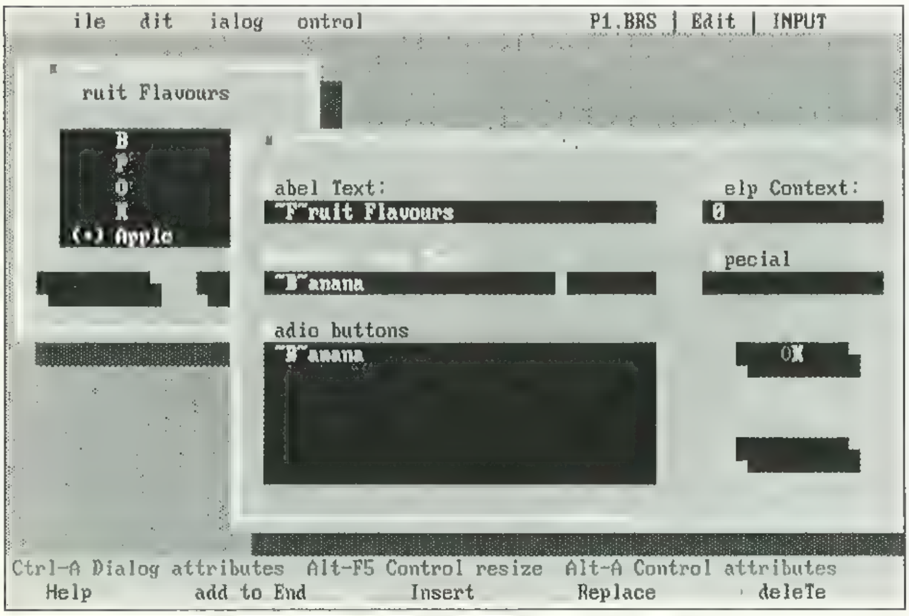

ile   dit   ialog   ontrol
P1.BRS | Edit | INPUT
ruit Flavours
B
P
O
K
(•) Apple
abel Text:
~F~ruit Flavours
elp Context:
0
pecial
~B~anana
adio buttons
~B~anana
OK
Ctrl-A Dialog attributes   Alt-F5 Control resize   Alt-A Control attributes
Help   add to End   Insert   Replace   deleTe

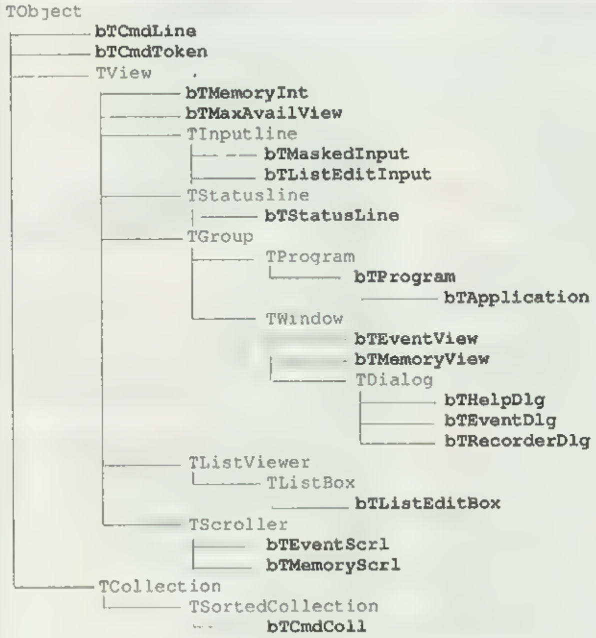
Figure 1 - The RESEDIT dialog editor

ADVANCED BUSINESS GRAPHICS

```
var
  DlgRes   : PDialog;
  RFile    : TResourceFile;
  R        : TRect

begin
  {Open Resource file}
  RFile.Init(New(PResStream, Init('DRES.BRS', stOpenRead, 1024)));
  {Get dialog box}
  DlgRes := PDialog(RFile.Get('DLG1'));
  DcckTop^.ExccView(DlgRes);
end;
```

Figure 2 - 'Shoehorning' a dialog control



```
TObject
  ├── bTCmdLine
  ├── bTCmdToken
  ├── TView
  │     ├── bTMemoryInt
  │     ├── bTMaxAvailView
  │     ├── TInputline
  │     │     ├── bTMaskedInput
  │     │     └── bTListEditInput
  │     ├── TStatusline
  │     │     └── bTStatusLine
  │     ├── TGroup
  │     │     ├── TProgram
  │     │     │     └── bTProgram
  │     │     │           └── bTApplication
  │     │     └── TWindow
  │     │           ├── bTEventView
  │     │           ├── bTMemoryView
  │     │           └── TDialog
  │     │                 ├── bTHelpDlg
  │     │                 ├── bTEventDlg
  │     │                 └── bTRecorderDlg
  │     ├── TListViewer
  │     │     └── TListBox
  │     │           └── bTListEditBox
  │     └── TScroller
  │           ├── bTEventScrl
  │           └── bTMemoryScrl
  └── TCollection
        └── TSortedCollection
              └── bTCmdColl
```

bTApplication	Replacement for standard TApplication. Supplied in two versions for debugging and release code.
bTCmdColl	Collection of command line token objects.
bTCmdLine	Command line parser.
bTCmdToken	Command line token.
bTEventDlg	Debug tool. Dialog box configures event viewer.
bTEventScrl	Debug tool. Displayable buffer of events
bTEventView	Debug tool. Window to display btEventScrl.
bTHelpDlg	Dialog which understands Turbo Vision's help system. Can detect F1 keystroke and gain control from modal dialogs.
bTListEditBox	Allows editing and ordering of string collections.
bTListEditInput	For editing an individual string in a bTListEditBox.
bTMaskedInput	Formatted version of TInputLine. Can be used to force numeric input etc.
bTMaxavailView	Debug tool. Displays largest available heap block on status line.
bTMemoryInt	Debug tool. Interior view provides details of a program's memory usage.
bTMemoryScrl	Debug tool. Interior view for scrolling through heap's freed blocks.
bTMemoryView	Debug tool. Window contains instances of bTMemoryInt and bTMemoryScrl.
bTProgram	Replacement for standard TProgam. Supplied in debugging and release versions.
bTRecorderDlg	Debug tool. Dialog box configures event recorder.
bTStatusLine	Together with bTProgram, allows the use of string resources for status line 'hints'.

Figure 3 - TVDT additions to the Turbo Vision hierarchy

TVDT

The heart of Blaise Computing's *Turbo Vision Development Toolkit* (henceforth TVDT) is RESEDIT, which is essentially one of those screen painter programs as supplied with 4GLs. Figure 1 shows it in action. The rest of this article deals with RESEDIT and its sister utilities and libraries. TVDT is available in separate editions for Turbo Pascal and C++; I'm told that these are nearly identical, but for the record this piece records my experiences with the Pascal version.

As you will have guessed, RESEDIT allows you to draw up a dialog box on-screen, filling it with your selection of buttons, radio buttons and check boxes, input (text) lines, static text and list viewers. Dialogs thus created may be saved in resource files, and can also be loaded back from resources and re-edited. RESEDIT is itself a Turbo Vision program, and is consequently very easy to use. Control objects may be dragged around, resized, copied to clipboard etc in the familiar Windows way. Double-clicking on a given control brings up a dialog of its attributes - this is how you change, for example, the static text component of a static text control. There is a Test mode, so that one can try out the new-born dialog, and an ingenious system for editing the Tab order (the order in which the Tab key moves around the different controls within the dialog).

The trouble with RESEDIT is that it can only handle classes that it 'knows' about, ie classes that are supplied with Turbo Vision. One often wishes to incorporate subclassed versions of controls into dialog boxes. For example, one might have developed a specialised version of `TInputLine` which accepted a COBOL `PIC` style format control string (in fact, TVDT includes just such a class, called `bTMaskedInput`). The `TListViewer` class *must* be subclassed to be of any use as it contains an abstract (== C++ 'pure virtual') method.

Shoehorning

Blaise has got round this using a fudgenique called *shoehorning*. It works as follows. Suppose you have overridden the `TButton` class, creating `TNewButton`. You create a dialog box using RESEDIT in the usual way, and add an ordinary button in the position that you want your `TNewButton`. Now bring up the button's attribute dialog and click on the shoehorn attribute (which, in fact, sets the high bit in the `options` field of the button's instance data - please see the TV manual's entry on `TView`). Save the dialog in a resource file and put to one side for the time being. Now

THE SOFTWARE CONSTRUCTION COMPANY

BORLAND

Borland C++ v.3.0	£189
C++ & Paradox Engine	£199
C++/App. Framework	£289
Turbo C++ v 3.0 New	£45
Turbo C++ for Windows	£79
Turbo Debugger & Tools	£85
Turbo Pascal v.6.0	£65
Pascal Professional	£129
Pascal for Windows	£99
ObjectVision v.2.0	£79
Paradox 3.5	£325
Paradox Engine	£189
dBASE IV 1.5	£325
dBASE IV 1.5 Upgrade	£80

UK AUTHORISED

COMMUNICATIONS LIBRARIES

AdComm	£190
Async Professional	£90
ASYNCH PLUS	£125
C ASYNCH MANAGER	£125
C Comm Tlkt Prof Ext DOS	£325
C Comms Toolkit Prof.	£195
C Communications Toolkit	£99
Essential Communications	£215
GET-IT	£99
Greenleaf Comm++	£129
ProBas Telecom Toolkit	£49
QuickComm	£99
SilverClip SPCS	£225
SilverWare C Asynch Lib.	£165
Solid Link with source	£325

DATA FILE HANDLERS

B+Tree Data Manager	£65
B-Tree Filer	£90
B-Tree Filer (multiuser)	£125
BTree/ISAM multi user	£115
BTree/ISAM single user	£75
Btrieve for DOS	£385
C++ Database Class Lib	£150
C-Data Manager w/source	£385
C-Index/II	£449
C-ISAM	£215
c-tree Plus	£385
c-tree with r-tree	£419
CBTREE	£129
CodeBase 4.5	£255
CodeBase 4.5 Upgrade	£95
db/LIB Database Prof Lib	£125
BFtrieve	£115
dp-MAX	£99
E-Tree Plus	£165
Essential B-Tree	£129
EZedit Unlimited Users	£195
Faircom Server	£190
Phase One	£319
Pinnacle Engine	£165
POET	£259
QBE/Phase 1	£255
Toolbox Professional	£835
Toolbox Special Edition	£515
Topaz - Multi-user DBMS	£99
r-tree Report Generator	£190

GENERAL FUNCTIONS

"C"erious Toolkit Plus	£65
3PX Extended Function Lib	£129
BASIC Development Tools	£65
Blackstar C Function Lib	£65
C TOOLS PLUS	£99
C Utility Library	£165
dBUtility Library	£100
FUNCky.LIB	£175
Grumpfish Library	£129
Object Professional	£125
ProBas Prog. Library	£115
ProBas Toolkit	£85
QuickPak Professional	£129
Rogue Wave C++ Class Lib.	£59
Soft.Clip (Net. version)	£195
Spindrift Library	£99
Spontaneous Assembly	£129
Super.LIB Prof. Network	£155
Super.LIB Professional	£129
SuperFunctions	£195
Tools.h++	£129
Turbo Professional	£89

GRAPHICS LIBRARIES

3-D Computerscape/source	£189
3-D MiniCAD w/source	£450

Microsoft

BASIC PDS 7.1	£215
Visual Basic	£90
C/C++ 7.0 Compiler	£215
C/C++ 7.0 Upgrade	£99
C 6.0 & Windows SDK	£315
COBOL 4.50	£420
FORTRAN 5.1	£199
MASM 6.0	£70
MS-DOS 5 Upgrade	£49
Mulimedia Dev. Kit	£215
Pascal Compiler	£135
QuickBASIC	£49
QuickC	£49
QuickC/Assembler	£75
QuickC for Windows	£90
QuickPascal	£49
Source Profiler	£39
Test for Windows	£179
Visual Basic Toolkit	£129
Windows 3.0	£60
Windows S.D.K.	£215

U.K. Authorised

ChartBuilder	£99
dGE	£195
EGA Toolkit	£159
Essential Graphics Chart	£905
Essential Graphics Kernel	£129
Flipper	£190
FontWINDOW	£99
Geograf Prof. Ed.	£210
GFORCE	£185
GFX Font & Icon Editor	£65
GFX Fonts & Menus	£99
GFX Fonts/Menus/Graphics	£165
GFX Graphics	£99
Grafmatic	£165
GrafPrint	£129
Graph Link	£129
GraphiC	£300
Graphix	£169
GraphPak Professional	£99
GSS Graphics Dev. Tlkt.	£515
GSS*CGM Metafile Interp.	£319
GSS*GKS Kernel System	£515
GX Developers Pak	£435
GX Effects	£129
GX Effects w/Source	£259
GX Graphics	£129
GX Graphics w/source	£259
GX Text	£99
GX Text w/Source	£195
Halo F/X	£195
Halo Image File Format	£165
HALO Professional	£385
INGRAF	£229
MEGAVDI	£579
MetaWINDOW	£190
MetaWINDOW/Plus	£255
MetaWINDOW/Premium	£385
NDP Plot	£200
PC Graphics Library	£385
PC-VDI	£255
PCX Prog Tlkt w/source	£389
PCX Programmer's Toolkit	£165
PHONTM-The Fontmaker	£255
ProGraphx Toolbox source	£259
Quick Geometry Library	£129
QuickWINDOW/C	£99
Turbo Halo	£65
TurboWINDOW/C	£99
TurboWINDOW/Pascal	£99
VID and DIG Graphics Lib	£259

GRAPHICS WINDOWS

Essential Graphics GUI	£195
Font-Tools w/source	£165
graphics-MENU	£165
graphics-MENU Pro Pak	£325
GSS XVT for DOS	£385
Object-Menu	£239
Object-Menu with source	£579
T-Windows with source	£190
TEGL Windows Toolkit II	£65
Zinc Interface Library	£129

COMMUNICATIONS & LANS

dB/LIB Network	£375
Net Lib	£195
Netware C Interface	£190
Network System Calls Clip	£190
Network Toolbox/N	£90

TEXT SCREENS

C-Worthy Library Source	£175
C-Worthy Upgrade to v.2.0	£125
C-Worthy/cwArchitect	£389

SHAREWARE FOR DEVELOPERS

ASM Library (20)	£40
C/C++ Library (68)	£66
dBASE/Clipper (87)	£100
DOS Utilities (37)	£40
Novell Netware (54)	£66
Turbo Pascal (38)	£40
Visual BASIC (11)	£40
Windows (64)	£100

(Number of 5.25 disks)

C-Worthy/cwArchitect/Srce	£559
Code Screen	£65
cwArchitect Source Code	£515
DataWindows	£255
HIM	£225
MEWEL Window System	£190
PANEL Plus II	£319
PANEL Plus II Util Source	£775
ProGraphx Menus source	£129
QuickWindows Advanced	£89
Vermont Views + Designer	£319
Vermont Views GraphEx	£99
Vermont Views QV	£165
Vermont Views with source	£575

386 COMPILERS & EXTENDERS

386 C Code Builder	£449
386|ASM/LinkLoc	£835
386|DEBUG	£129
386|DOS-Extender Dev. Kit	£319
386|VMM	£190
C Network Compiler/386	£645
C++ Developer's Edition	£399
EDU-16	£319
F77L-EM/32 w/OS/386	£899
Fortran 77 386/486	£645
LinkLoc	£515
Metaware High C 386/486	£645
MetaWare Pascal 386/486	£579
NDP C++	£545
NDP C-386	£545
NDP C-486	£750
NDP C-SX	£350
NDP Fortran-386	£545
NDP Fortran-486	£750
NDP Fortran-SX	£350
NDP Pascal-386	£545
OS/286 Developer's Kit	£449
OS/386 Developer's Kit	£449
Smalltalk/V 286	£129
SVS C'ANSI C	£775
SVS Fortran-77/386	£965
SVS Pascal/386	£775
WATCOM C 8.5/386 Prof	£515
WATCOM Fortran 77/386	£515

OTHER C COMPILERS

C-terp	£195
Instant C	£319
Metaware High C DOS x86	£319
TopSpeed C - Professional	£189
WATCOM C 8.5 Professional	£319

DATABASE MANAGEMENT

Alpha Four	£345
Clarion Personal Dev. 2.0	£79
Clarion Professional Dev.	£679
PALCOM Network	£645
PALCOM Single User	£319

xBASE

Clipper 5.0	£325
FoxBASE+	£295
FoxPro	£495
FoxPro/LAN	£695
Quicksilver	£389

OBJECT-ORIENTATED

Smalltalk/V	£85
Smalltalk/V 286	£129
Smalltalk/V Windows	£325

PASCAL

MetaWare Prof. Pascal x86	£385
Stony Brook Pascal+	£190
TopSpeed Pascal - Prof.	£109
TopSpeed Pascal - Std.	£109

SQL

Faircom SQL Server	£319
FirstSQL	£319

UPGRADE TO MICROSOFT C/C++7.0

Microsoft C (any)	£99
Borland/Turbo C++ (any)*	£99
Zortech C++ (any)*	£99
Topspeed C++ (any)*	£99

**REQUIRES PROOF OF PURCHASE BEFORE 1ST FEBRUARY 1992 . ALL PRICES + VAT*

*All prices exclude V.A.T. U.K. carriage included. VISA or ACCESS accepted. Credit to approved accounts. New **Toolbox 5.0** catalogue now available, call. Please check prices when ordering and specify disk size. 3.5" now shipped as standard.

1 THE MALTINGS, GREEN DRIFT, ROYSTON, HERTS. SG8 5DB
TELEPHONE (0763) 244114 FACSIMILE (0763) 244025

CIRCLE NO. 826

modify the dialog-loading code along the lines shown in Figure 2.

Shoehorning does mean that one is not totally stuffed when using derived classes, but I have to admit to just a little disappointmentthat the lads at Blaise hadn't devised some fiendishly clever way of making RESEDIT recognise user-derived objects.

The trouble with RESEDIT is that it can only handle classes that it knows about

As well as dialogs, RESEDIT can cope with two other types of resource. The Menu Bar editor allows one to create and manipulate a standard Turbo Vision drop-down menu bar. This will be especially welcome to fellow Pascalians, who can remove the hard-to-maintain hand-coded equivalents which, thanks to huge numbers of nested function calls, look more like LISP than Pascal. String list resources provide a simple way of keeping your strings separate from the application code, in order to make it easy to create the French language version of your masterwork.

Extra Objects

There is another plank to TVDT, in the form of a library of objects and functions. The objects are all additions to the Turbo Vision hierarchy, and are outlined in Figure 3. The most interesting are probably the two `bTApplication` classes.

The version of `bTApplication` in the unit (= C++ library) file bApp provides a little bit of extra housekeeping to ease the handling of resources created with RESEDIT. The alternative `bTApplication`, the one in unit file BetabApp, adds a raft of debugging aids.

A 'β' item appears on the left hand side of the menu bar. Clicking on this drops a menu which offers event monitoring (much like the 'Spy' utility in Windows), recording and playing back of events to/from a file (so that it is possible to record a debugging session to a certain point and play it back as required), a memory viewer (shows PSP, current stack and heap information, including the current free block list), a MaxAvail viewer (much like the one bundled with Turbo Vision), the ability to switch between 25 and 43/50 line text mode, and a DOS shell.

There's some other library material which deserves a quick mention. `bTMaskedInput`, briefly alluded to above, is a derivative of `TInputLine` which lets you constrain user input with a picture mask (eg `'#9999'` forces a numeric or sign character, followed by four numerics). `bTCmdToken`, `bTCmdColl` and `bTCmdLine` allow simple parsing of command lines - the objects can tell the difference between switches and other tokens. There's also lots of little utility functions like `bFileExist` and `bUpCaseStr` which are ok, but which you have probably already written for yourself.

Oddments

Porting your TV applications to Windows? The RC-1 utility (stands for 'RC inverse'), also bundled with this kit, sucks in Turbo Vision resource files and spits out equivalent scripts for the Windows RC compiler. This doesn't mean that you will be up and running in 3.1 in five minutes but, as a colleague pointed out, designing dialog boxes is a time-consuming matter.

I should mention the manual which, apparently like all Blaise's efforts, is produced in exactly the same style and typefaces as a Borland manual. Everything that one might reasonably expect is included (except, perhaps, a class hierarchy like the one in Figure 2): good tutorial, reference section, appendices on trouble shooting and tips, and an index.

Conclusion

I think that the Turbo Vision Application Framework attracts less than its fair share of attention, so I was pleased to see Blaise taking it seriously. This toolkit is produced very much in the same spirit as the Borland product - there are lots of interesting bits to be found in here, and everything is a lot less unwieldy than the mega multi-platform, multimedia 3D add-ons you get for full-blown GUIs. I commend it to fellow TV addicts.

Blaise Computing's Turbo Vision Development Toolkit *is available from various UK dealers. The review copy was supplied by System Science (071 833 1022), which quotes £89.00 ex VAT for both C++ and Pascal versions.*

Crash Protection in C++

*Nikki Locke shows how to diagnose system crashes in C++ programs **before** they happen, without the aid of a debugger.*

I have recently had some very unpleasant experiences trying to port a text mode user interface toolkit from Zortech C++ to Borland Turbo C++. My problems arose from the fact that my code was not as portable as I thought. Like most C++ compilers, neither Zortech C++ nor Borland Turbo C++ are fully compliant with *The Annotated C++ Reference Manual* (the ARM for short), the 'bible' for the C++ language. I found that code with which Zortech was completely happy was, in fact, full of errors!

This, in itself, was a pain, but one I could live with. I went through my code, correcting the many errors, and compiling with the Turbo compiler. When this mammoth task was complete, I thought testing and debugging the code would be relatively simple. This was **not** the case!

Unfortunately, there were still many errors in the code which gave rise to classic 'undefined behaviour'. Most C and C++ programmers are surely conversant with the term - it means anything from 'running perfectly' through 'hanging the system' or 'rebooting the machine' to 'trashing the Non Volatile RAM'. The worst form of such errors is when a virtual function is called for some pointer which does not point to a valid object. Because the address of the function is taken from the vtable, whose own address is supposed to be inside the object, the result is equivalent to a call to a random address.

Debuggers

I had become used to the excellent Zortech debugger, which allows tracing through generated code in assembler mode. This is an extremely useful feature when you want to know what is **really** going on, but it was sadly lacking in the ancient version of the Turbo debugger I was using. To add insult to injury, I found code that worked perfectly when compiled with debugging information on, but failed totally when recompiled with debugging information off.

In any case, debugging in assembly mode is fine if you know roughly where the problem is - it is not much help if your program hangs the machine somewhere deep in the list of calls to static constructors, as it can take ages to trace through the list before you find out what is going wrong.

Validation of objects

What I needed was a simple, cheap way of checking the validity of any object. The obvious answer was to have a private validity checking member of the object that is set to some known value when the object is constructed, and reset when the object is destroyed.

```
// Example class derivation
class Base : public Protected {
private:
  // Overridden in derived classes
  virtual void v_virtfunc ();
public:
  Base ();
  virtual ~Base ();
  // Calls virtual after checking.
  void virtfunc ()
    { VALIDATE (this); x_virtfunc (); }
  };

class Derived : public Base {
private:
  // Override Base function
  virtual void x_virtfunc ();
public:
  Derived ();
  ~Derived ();
    // No need to override virtfunc
  };
```

Figure 2 - Virtual functions with validity checking

```
// Header file : protect.hpp

#ifdef NOPROTECT
// No-overhead version for use when
// debugging is complete
class Protected {
public:
  int valid() const { return 1; }
  };

#define VALIDATE(p) ((void)0)

#else
// Validity-checking version

#include <assert.h>
// Use standard "assert" package

#define VALIDATE(p) assert(p->valid())
// Check a given pointer points
// to a valid object

class Protected {
private:
  // The validity checking pointer
  Protected *me;
public:
  Protected();
  ~Protected();
  int valid() const {return (this == me);}
  };

#endif

// Implementation file : protect.cpp
#include "vobject.hpp"

#ifndef NOPROTECT

Protected::Protected ()
: me(this)     // Set up validity pointer
{
}

Protected::~Protected ()
{   // Ensure not deleted twice !
  VALIDATE(this);
  me = 0;  // Clear down validity pointer
}
```

Figure 1 - Validity checking base class

I created a new base class, called `Protected`, to hold the validity checking members, so that I could subclass it for all classes with virtual functions. I wanted the checking member to have a unique value for each different object, and I wanted a value which was unlikely to be duplicated accidentally. I therefore made the validity checking member a pointer to `this` - I called it `me`.

It was a simple matter to add an inline call `valid` to check whether `me` was indeed equal to `this`. I also added a macro `VALIDATE(ptr)` which aborted the program if the object pointer `ptr` was not valid. Note that a macro was used, rather than an inline function, so that the C `assert` macro would return the line number of the function calling `VALIDATE`, rather than the line number of the inline function. Figure 1 gives a minimal example demonstrating the technique.

Deriving all classes from the `Protected` class is all very fine in development code,

CONGRATULATIONS. YOUR NEXT CUSTOM APPLICATION IS ALREADY HALF FINISHED.

The graphical revolution made computers far easier to use, but infinitely more difficult to program. That's why the industry is so intent on object-oriented systems, where powerful applications are created in a fraction of the time.

For most companies, just the first effort is still at least three years away. But NeXT™ has a proven system that works today.

The NeXTstep™ environment is an object-oriented world right down to the system level, offering tools you've never had before.

NeXTstep's Application Kit™ gives you the objects common to most programs, ready to use without any modification. It even includes objects for faxing, choosing fonts/colours and interapplication communication – all with standard dialogues.

With the Interface Builder,™ you can use the mouse not only to place and resize your objects, but to change their attributes and define interaction – all without writing code.

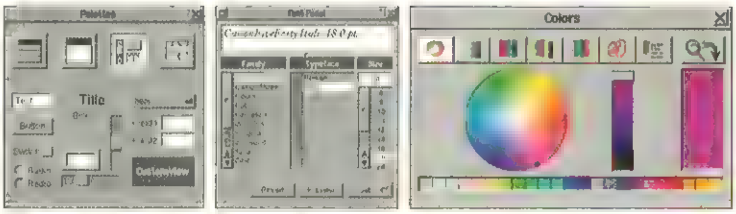
For even greater power, the NeXTstep object library is wide open. That is, you can customise any object's behaviour through the use of subclassing or create new objects.

Because you can reuse objects, there's no need to reinvent the wheel with every new project. And to update, you need only to modify a single object.

With NeXT workstations, custom software is both intuitive and consistent, offering stunning graphics and access to all system services.

Further, it integrates seamlessly with other applications, custom and commercial.

For more information on NeXT computers and our developer programme, please complete the coupon or telephone 081 565 0005.



The NeXTstep development environment delivers all the benefits of a true object-oriented system. Sophisticated application objects are all provided, along with the power to subclass and create exactly the objects you need

CIRCLE NO. 828

I'd like to receive more information on NeXT □

Please complete and return to: NeXT, 1 Heathrow Boulevard, 286 Bath Road,
West Drayton, Middlesex UB7 0DQ.

Name:

Company:

Address:

County:

Telephone: Post code.

.EXE 4/92

Now for WINDOWS 3 – *still only £495*

Personal-SELECT front-end
CASE tools for SSADM and Yourdon

True multi-user for an additional £995 per network!

More than 2500 professional software developers in blue chip companies know that SELECT provides first class PC-based front end CASE tools.

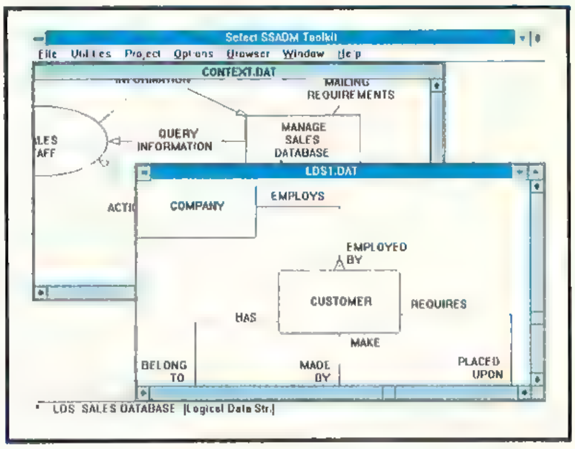
What is more the SELECT tools are highly rated by independent experts:

'the best to date at any price' Software Development Monitor.

'Exceptional value for money' Ovum Ltd.

Now they are on Windows, what are you waiting for?

Contact our Sales Department on 0242 226553 for more details or to place your order.



SELECT
Software Tools

SELECT Software Tools Ltd, Ullenwood Court, Ullenwood, Cheltenham, Gloucestershire GL53 9QS, Tel: 0242 226553, Fax: 0242 251491

CIRCLE NO. 829

BOSTON SYSTEMS OFFICE/TASKING

16 Fernhill Road, Farnborough, Hants GU14 9RX, England

REAL ENGINEERS WOULDN'T GIVE A XXXX FOR ANY OTHER REAL TIME EXECUTIVE

If you want a fully pre-emptive real time executive with fast context switch time, dense code size and fast primitive execution time you need BSO/Tasking's Real Time Craft.

If you are programming the 80XX6 series, the 68XXX series, the AMD29000 or other 16/32 bit devices and you are programming in C, Pascal, Fortran or Assembler you need BSO/Tasking's Real Time Craft.

If you want a product that comes with one years free technical support from a worldwide manufacturer of software tools you need BSO/Tasking's Real Time Craft.

For a FREE information pack contact 0252-510014 or write to the address at the top of this advertisement. Alternatively write in number **60** on the reply card.

Real Time Craft. THE Real Time Executive

CIRCLE NO. 830

but would result in unacceptable overhead in production code. Therefore I provided an alternative versions (enabled when NOPROTECT is defined). This alternative version is a class with no members, no constructors and no virtual functions, giving zero (or nearly zero) overhead.

In an ideal situation, I wanted to call VALIDATE before every virtual function call. The most elegant way of doing this is to make all virtual functions private, and to provide inline, non-virtual, helper functions which call VALIDATE before calling the private, virtual function. These helper functions only need to be included in the first ('basest') class that defines a particular virtual function. They thus give the side benefit of identifying exactly where in the inheritance tree a virtual function is first defined.

Because private virtual functions cannot be called from outside the class, accidental calling of virtuals without checking the validity of the object is prevented. However, C++ **does** allow the overriding of private virtuals in derived classes, so full functionality is retained.

Figure 2 illustrates how virtual functions can be defined in this way.

Startling results

Once I had added this new class to my code, and inherited it in all my base classes, I made some startling discoveries.

I had lots of code which depended on the order of calling static constructors and destructors in different object modules. Zortech C++ always calls the constructors in the same order - starting with the last module

```
// Header file : protect.hpp

// Protection for classes. A class which
// registers itself into a debugging list
// (one for new'ed items, one for static
// & automatic) which can be dumped
#include <stdio.h>

#ifdef NOPROTECT
// No checking, no overhead version

class Protected {
public:
  int valid() const { return 1; }
  static void dumpstack(FILE *f = 0) {}
  static void dumpheap(FILE *f   0) {}
  static void dumpall(FILE *f = 0) {}
  // prevent linking mixed versions
  static char off();
  };

#define FUNCTION(ignore) ((void)0)

#define VALIDATE(p) ((void)0)

static const char protectionCheck =
                    Protected::off();

#else

#include <assert.h>

#define VALIDATE(p)  \
         (p->valid() ? \
         (void)0 : \
         (Protected::dumpall(), \
                      assert(0)))

class Protected {
private:
  Protected *me;
  // Linked list next pointer
  Protected *next;
  int newflag; // != 0 if created with new
  // Last return value from operator new
  static void *lastnew;
  // Size of last new'ed item
  static size_t length;
  // Static/automatic list
  static Protected *pstack;
  // New (heap) list
  static Protected *pheap;
  // Print details of this object
  virtual void v_dump(FILE *f) const;
public:
  Protected();
  ~Protected();
  int valid() const {return (this == me);}
  int isnew() const { return newflag; }
  void dump(FILE *f) const
    { VALIDATE(this);
      v_dump(f ? f : dumpfile); }
  void *operator new(size_t);
  static void dumpstack(FILE *f = 0);
  static void dumpheap(FILE *f = 0);
  static void dumpall(FILE *f = 0)
    { dumpstack(f); dumpheap(f); }
  // Prevent linking mixed versions
  static char on();
  static FILE *dumpfile;
  },

static const char protectionCheck =
                     Protected::on();

#endif

// A Protected item with a name
class NamedProtected : public Protected {

#ifndef NOPROTECT
  // New version of dump prints name
  virtual void v dump(FILE *f) const;
#endif
protected:
  const char *namestring;
public:
  NamedProtected(const char *name);
  const char *name() const
    { return namestring; }
  };

#ifndef NOPROTECT

// A named function tag with file
// and line number
class Function : public NamedProtected {
  virtual void v_dump(FILE *f) const;
  const char *filename;
  unsigned lineno;
public:
  Function(const char *name,
           const char *file,
           unsigned line);
  const char *file() const
    { return filename; }
  unsigned line() const
    { return lineno; }
  };

#define FUNCTION(name) \
    Function(name,__FILE__,__LINE__)

#endif

// Implementation file : protect.cpp
#include "protect.hpp"

#ifndef NOPROTECT

void *Protected::lastnew = 0;
size_t Protected::length = 0;
Protected *Protected::pstack = 0;
Protected *Protected::pheap = 0;
FILE *Protected::dumpfile = stderr;

Protected::Protected()
: me(this)
{
  if(lastnew <= this &&
     (char *)lastnew+length>=(char *)this)
    {  // was created using new
    next = pheap;      // add to heap
    pheap = this;
    newflag = 1;       // mark as new'ed
    }
  else
    {  // was static or automatic
    next = pstack;     // add to stack
    pstack = this;
    newflag = 0;  // mark as not new'ed
    }
}

Protected::~Protected()
{ // Ensure not deleted twice !
  VALIDATE(this);
  if(isnew())
    {
    // Must walk heap until we find this
    Protected **prevptr = &pheap,
                *n = pheap;

    while(n != this)
      {// this not found in heap - error !
      assert(n);
      prevptr = &n->next;
      n = n->next;
      }
    // Now remove from heap list
    *prevptr = this->next;
    }
  else
    {
    // We assume that items are destroyed
    // in reverse order to their creation.
    // If this is not the case for your
    // compiler, the code for the heap
    // (above) should be used instead.
    assert(this == Protected::pstack);
    pstack = this->next;
    }
  me = 0;
  next = 0;
}

void Protected::v_dump(FILE *f) const
{
   if(f)
    fprintf(f,"Protected@%p from %s\n",
            this,
            isnew() ? "heap" : "stack");
}

void *Protected::operator new(size_t len)
{
  // Register all calls to new
  length = len;
  return (lastnew = ::operator new(len));
}

void Protected::dumpstack(FILE *f)
{
  if(!f)
    f = dumpfile ? dumpfile : stderr;
  fprintf(f,"\nStack :-\n");
  for(Protected *p = pstack; p;
      p = p->next)
    if(p->valid())
      p->v_dump(f);
    else
      fprintf(f,"Invalid object @%p\n",p);
}

void Protected::dumpheap(FILE *f)
{
  if(!f)
    f = dumpfile ? dumpfile : stderr;
  fprintf(f,"\nHeap :-\n");
  for(Protected *p = pheap; p;
      p = p->next)
    if(p->valid())
      p->v_dump(f);
    else
      fprintf(f,"Invalid object @%p\n",p);
}

void NamedProtected::v_dump(FILE *f) const
{
  if(f)
    fprintf(f,"%s@%p\n", name(), this);
}

Function::Function(const char *name,
                   const char *file,
                   unsigned line)
: NamedProtected(name), filename(file),
                        lineno(line)
{
}

void Function::v dump(FILE *f) const
{
  if(f)
    fprintf(f,"Function %s@%p in file %s \
at line %u\n",
      name(), this, file(), line());
}

char Protected::on() { return 1; }

#else

char Protected::off() { return 0; }

#endif

NamedProtected::NamedProtected
                     (const char *name)
: namestring(name)
{
}
```

Figure 3 - Stack trace and heap walk added

linked, and finishing with the first module linked. Destructors were always called in reverse order to constructors. My rather old version of Turbo C++ called constructors in an entirely different order, and called destructors (in different modules) in the same order as constructors, rather than in reverse order. I understand that this has now been changed, but the ARM does not guarantee anything about the order of constructors or destructors between different modules, so both compilers are conformant in this respect.

I also found some other code that used objects after they had been deleted, or before they had been constructed.

My new `Protected` class identified both these types of problems instantly, without any heart-stopping 'undefined behaviour'.

I sent my code to a friend who has a full V2.0 of Borland C++, and it found a serious bug in the compiler. For those interested in these things, deriving a class from two base classes, the second of which had a virtual function, and the first of which didn't, caused the derived class to be constructed *without calling the base class constructors.*

With protection disabled, this gave rise to some very interesting inexplicable behaviour. Enabling protection instantly highlighted the problem !

More information

Although this technique certainly reduces unexpected system crashes, the information returned is somewhat minimal - the source file and line number if the function that called `VALIDATE`. Usually, this was the automatic call of `VALIDATE` in the inline helper function, in a header file.

I found myself wanting a stack trace, so that I could find out which functions had attempted to use the invalid object. I also wanted to trace memory leaks - the Zortech debugger will do this for you automatically, but most other debuggers won't help at all.

I found I was able to provide both these facilities with simple extensions to my `Protected` class.

First, I added operator `new` to the class in order to detect whether a given object was created on the stack or the heap. This gave me the added advantage that I could query any `Protected` object to see how it was created. This technique is worth noting, as it is often necessary to know whether certain classes were constructed with `new` (so you know whether to `delete` them).

Then I added code to add the object to one of two linked lists (stack or heap, depending on whether it was created with `new`) in the constructor, and remove it from the list in the destructor. Note that both automatic and static objects are listed in the 'stack' as I have not found a portable way to distinguish between them.

Finally, I added a virtual function to dump information about the object to a `FILE`, and static functions to dump the stack and the heap. Note the use of an old, C-style, `FILE` pointer for the dump, rather than using a new C++ stream. This is to avoid problems where a dump is requested either before the standard streams have been constructed, or after they have been destroyed.

The code in Figure 3 contains a definition of the classes.

I created a subclass of `Protected`, called `NamedProtected`, which retained a pointer to its name, and printed the name when dump was called. I found this useful, as many of the objects in my toolkit already had a name which I could pass on to the `NamedProtected` constructor, giving a more useful dump.

Finally, the class `Function` was derived from `NamedProtected`, and a `FUNCTION(name)` macro defined. I only had to place a `FUNCTION` macro call at the start of important functions, and they were automatically added to the stack trace.

The use of the `NOPROTECT` macro to turn off all protection (and any overhead associated with it) could give rise to a new problem - if two modules, one compiled with `NOPROTECT` defined, the other not, are linked together, strange things will start to happen. One module expects `Protected` to have a virtual function, the other doesn't. This can give rise to a whole new order of bugs.

In order to prevent this happening, I included a different static function in the `Protected` class - `on` if `NOPROTECT` wasn't defined, `off` if it was. I also included a `static const` in the *header* file, initialised to the return value of the appropriate function, so that every module that includes the `Protected` header file references one of these functions. When many modules are linked together, a link error will occur if some use `NOPROTECT` and some don't.

Using the data dumps

With a modification to the `VALIDATE` macro to dump the heap and stack if the assertion failed, and a static member `Protected::dumpfile` to handle the output, I now had a fully automatic stack and heap dump whenever something went wrong.

In order to trace memory leaks, it is only necessary to call `Protected::dumpheap` at the end of the main function to see which objects allocated with `new` have not yet been deleted. Obviously, memory allocated with `new` during static object constructors will not have been deleted, but you can usually account for these items by studying your code.

Of course, if you know the order in which static destructors are called for your particular compiler, you can create a static object whose destructor does this check, and save yourself some work.

More possibilities

Once all your significant classes are derived from `Protected`, there are numerous other enhancements you can make to the simple validity checking shown in this article. For example, the `delete` operator could be overridden to ensure only items allocated with `new` are deleted. Profiling information (function call times, etc) could be accumulated using the `Function` class.

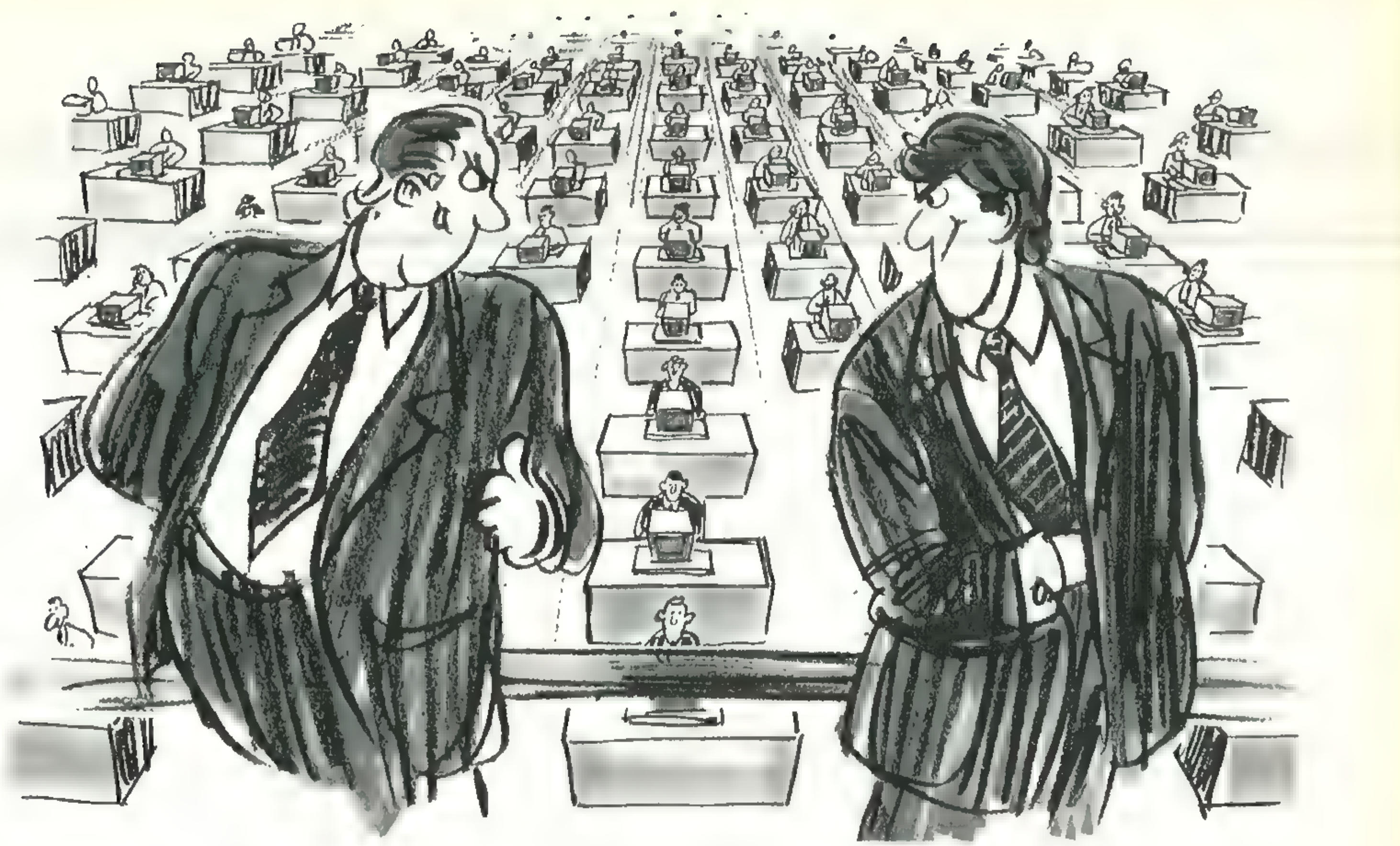
However, it is easy to get carried away with this sort of thing. Re-inventing Smalltalk in C++ is **not** a terribly good idea! I would not recommend trying to store type information in the `Protected` class, and querying the type of an object in your code, for instance.

It is far better to limit the `Protected` class to providing debugging and/or tuning information, so that any overhead can be switched out in production code.

EXE

Nikki Locke is a freelance consultant working in C and C++ on DOS, OS/2 and UNIX. He has been programming in C since 1980 (anyone remember BDS C under CP/M?), and has been using C++ since 1989. He can be reached via email at `nikki@cix.compulink.co.uk`, *by telephone on 0691 670318 or FAX on 0691 670316.*

This code is available free to anybody who sends in a disk. Please follow the instructions on Page 1 Column 1, and mark your envelopes 'VALID'.

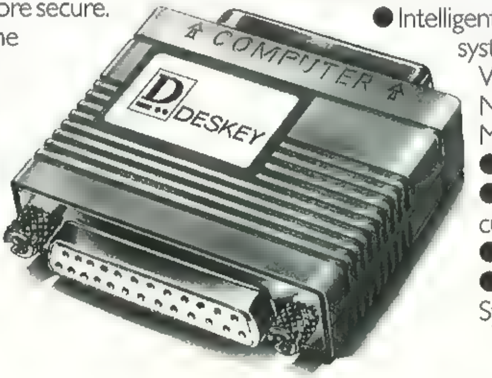


"You only need to buy one copy! They don't protect with DESkey."

DESkey – Software Protection from the inventors of the intelligent Dongle.

We have more experience and expertise in the design of software protection modules (dongles) than any other company in the UK. We sell products designed in house by our own engineers. More than 13 years experience in the design of dongles have gone into our current product range.

- DESlock© automatic .EXE or .COM file encryption system. No need for source or .OBJ files. Takes only a few seconds to protect your software – *now available for OS/2 and Windows.*
- 'Seedable' Pseudo Random Number Generator. Software and data encryption could not be more secure.
- Through Encryption. Data can be fed into the DESkey for on-line encryption. This keeps the encryption key hidden.
- Memory. Up to 240 bytes of memory split into 'Public' and 'Private' sectors. The 'Public' sector may be read from and written to at will. The 'Private' sector may be read at will, but writing requires your customer specific password.
- Down Counter. You program into the DESkey the number of times the program will run before stopping. This gives you the ability to sell 'goes' of your software rather than an open licence to use it forever – or even to send a fully working demonstration copy that will stop working after say, 10 goes.
- Variable Response Algorithm. This feature is similar to the well known 'public-key' algorithm and works in conjunction with an algorithm on the host computer. Makes any attempt at software emulation impossible.
- Secure Memory Read. Even if the same memory data is read repeatedly, the data returned from the DESkey never appears the same – this also makes the DESkey impossible to emulate.
- Parallel, Serial or Bus versions available.
- Intelligent Serial devices suitable for any operating system such as Unix, Xenix, OS/2, DOS etc. Works on any hardware including PC Networks, Mini Systems or even Mainframes.
- Free evaluation Kits.
- Guaranteed Exclusivity to all our customers.
- Fast Order Turn-round.
- Sold only through Data Encryption Systems Limited.



Edbrook House,
Cannington, Bridgwater,
Somerset, TA5 2QE

Telephone 0278 653456
Fax 0278 653300

D..
Data Encryption Systems Limited

FAST
Federation Against Software Theft

Data Encryption Systems
is a member of FAST

CIRCLE NO. 831

Malloced

Remember the last time your code inexplicably 'bombed-out'. Cliff Saran points a finger at the culprit and searches for a better solution.

A C programmer needs `malloc` like a fish needs water. But if we've been handed a loaded gun (he says, switching metaphors rapidly), what can we do to prevent us from shooting ourselves in the foot? The standard C run-time library provides several memory management routines, but it lacks even the barest of debugging facilities. There are many reasons why `malloc` could fail, but when it does, it returns `NULL` and an error code in the `errorno` global variable. The four third-party Heap Management libraries in this article provide various debugging facilities. They also boast increased performance and reduced fragmentation. How do these third-party Heap Management libraries compare to the memory management routines in the C run-time library? I have chosen to use the Borland C++ V3.0 run-time library exclusively as a baseline (at the time of writing, Microsoft C V7.0 had not been released, so it would not have been fair to include its run-time library in these benchmarks). To keep everything consistent, all the benchmarks were compiled with Borland C++ V3.0 using Large memory model and optimisation switched off. Figure 1 shows the results of the benchmark (a description of the benchmark is given in the box below).

Dolphin

The Dolphin Far Memory Manager and Debugging Library provides a total of 62 routines. This library isn't a plug-in replacement for `malloc`. Nor does it provide access to EMS/EMS/Virtual Memory. It contains instead it's own heap management routines which provide comprehensive debugging facilities.

Unlike the run-time library which will only clear a block of memory (ie using `calloc`), Dolphin lets you set up a newly allocated block with a multi-byte value eg

```
char* s[] ={"Hello World"};
char *p;
int Error;
p = memalloc(50, &Error);
mem_init(p, 10, s);
```

The above code will allocate a 50 byte block pointed to by p and fill the block with the string 'Hello World'.

Pascal has been praised for providing array-bounds checking. Dolphin goes one step further and offers C programmers the security of array-bound checking at run-time. Any attempt to use an index out of the permitted range is trapped. The array index base can be changed from zero to any integer value (negative as well as positive). However, in order to use these features, Dolphin has provided special routines to index the array (ie if you want array-bounds checking, you can't index the array as `ch = Table[i][j]`).

```
char **p;
char ch;
p = array_get(&Error, Table,
              i,j);
ch = *p;
```

Many of the Dolphin routines return an error code and this is where the strength of the Dolphin library lies. It checks for 28 unique errors including null and invalid pointers, attempting to free an already free block, heap corruption and array index out of range errors. It is possible to set up Dolphin such that it causes the program to terminate when an application generates one or more of these error codes. There is also a Heap Log which writes a report of all operations that occur on the Heap.

MoreHeap

MoreHeap is a binary compatible replacement for the memory management routines in the C run-time library so an application only has to link with the MoreHeap library in order to use it. MoreHeap offers additional features not found in the run-time library and it is these that I will concentrate on.

First, MoreHeap gives you access to the video memory on EGA/VGA cards, providing up to 64 KB extra space. The problem with this technique is that it limits you to text-only applications and it cannot be used in conjunction with any program that writes directly to the screen memory.

Next, MoreHeap can take advantage of Upper Memory Blocks. This requires the use of an XMS device driver such as HIMEM.SYS. With the arrival of MS-DOS 5.0 this use of the UMBs is made redundant since it is possible to tell DOS to use the Upper Memory Blocks itself. However, machines running an application under a previous version of DOS can still use MoreHeap to access the UMBs.

Manufacturer	BC++ V3.0 Borland	C-Heap Library Technologies	Dolphin Dolphin Software	MoreHeap SeaBreeze Software	XMem Kandu Inc
Time to Malloc	0.66	0.16	2.47	1.26	0.16
Time to Calloc	0.93	0.16	11.70	36.15	0.60
Time to HeapWalk	0.00	84.07	4.84	0.11	-
Time to Alloc Virtual Handle	-	9.87	-	3.30	9.45
Time to Deref Virtual Pointer	-	0.0	-	4.12	27.42
Time to Free Virtual Handle	-	0.0	-	1.37	0.05
Compatible with RTL	BC	BC/MSC	x	BC/MSC	x
Compiler support	BC	BC/MSC	BC/MSC	BC/MSC	BC/MSC
Number of Error codes	0	0	27	0	6
Access to Memory Above Conventional RAM	x	EMS XMS	x	EMS XMS Video	EMS XMS
Heap Log	N	N	Y	Y	N
Number of Function	29	>500	62	48	36
Price	-	$399	$99	$135	£145

- -Does not apply *x -Not supported*

Figure 1 - Heap Libraries Benchmark

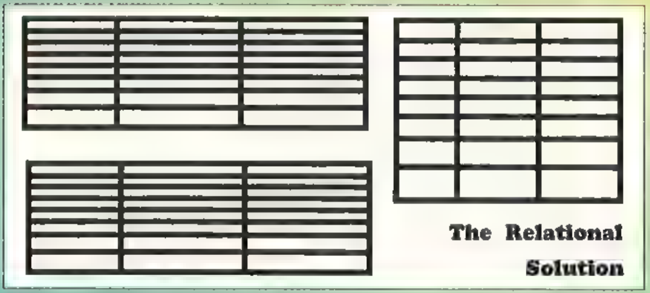
Does the Real World fit your Database, Or your Database fit the Real World ?

Problem :

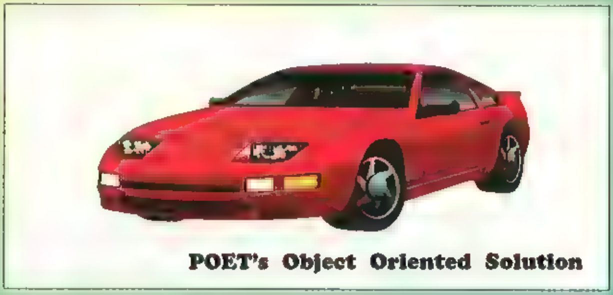
To Efficiently Store Complex Real World Objects in a Database.

STOP PRESS!

POET from BKS Software wins product excellence award at Software Developers 92, co sponsored by Computer Language Magazine and Jolt Cola.
.......San Francisco
26/2/92.........END.



The Relational Solution



POET's Object Oriented Solution

"Announcing P.O.E.T, the first true Object Oriented Database system for DOS, Windows, UNIX and NeXT, with prices starting from just £249"

The Affordable Database Solution.

Up until now, object oriented DBMSs have typically only been available on high end UNIX machines, with prohibitive entry prices of $20,000++. Now you can unleash the power and potential of a true OODBMS in your DOS or Windows application, at prices starting at £249 +VAT.

Object Oriented DBMS v Relational

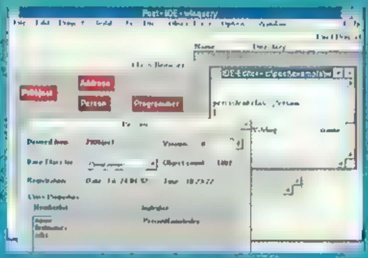
Relational Database (RDBS) and B-Tree/ISAM systems require that data be portrayed as a series of two dimensional tables. These systems were a real breakthrough in their time, but software developers are rapidly learning that life is not a series of two dimensional tables.

For example, a CAD program might have an array of shapes, or a DTP program might model a page as a series of frames which might contain bitmaps, paragraphs or vector drawings. Conventional databases have a fixed set of data types. And the only way they group data is in a table.

All trademarks acknowledged

Object-oriented programs allow us to declare new data types as needed. And POET stores the objects we create *as* objects. This one-to-one correspondence of the database design to its physical storage can realize access gains of over one hundred fold compared with relational DBMSs. POET has the speed while keeping its kernel small; the DOS runtime is under 256Kb.

POET includes complete support for C++ inheritance and encapsulation, class dictionary with class versioning, optimized indexing and querying, object caching, complex operators and containers. POET has what it takes.



Persistent Object Extended database Technology

POET took over 20 man years to develop by leading German Software house BKS. In non German speaking Europe it is distributed by Silicon River Limited, the same team that first popularized C and C++ in the UK by aggressively pricing Zortech C/C++.

What Should You Do?

This is the powerful, flexible database technology you have been awaiting. Now it is not only available, but affordable. If you will be developing a database application on any of the platforms below, then please call for a free Technical Overview of POET, the object oriented DBMS engine for the 1990s.

Platforms Supported:

Borland C++ (DOS or Windows)
Microsoft C/C++ 7.0 (DOS or Windows)
Zortech C++ 3.0 (DOS or Windows)

LPI C++ (UNIX 386 Systems)
HP C++ Softbench
GNU C++ (NeXT Step)

For your free POET Technical Overview call

081 317 7777

Silicon River

Silicon River Ltd 58-60 Beresford Street, LONDON, SE18 6BG
Voice 081 317 7777 Fax 081 316 7778 BBS 081 317 2310

CIRCLE NO. 851

MoreHeap can also use the HMA - if you are using DOS 5.0 with DOS loaded high, the manual suggests that by removing DOS=HIGH from CONFIG.SYS you will obtain more heap space (since MoreHeap can use all 64 KB of the HMA while DOS only uses about 40 KB).

So far, MoreHeap has only managed to scrape a few kilobytes here and there to extend the heap. With MS-DOS 5.0, the above techniques are no longer required - DOS can use the HMA and UMBs, giving applications more free conventional memory. To get the most Heap, you have to use MoreHeap's virtual memory manager. This enables MoreHeap to use extended memory and hard disk for swapping blocks in and out of physical memory addressable by the application. As conventional memory becomes full, MoreHeap will swap data back and forth to extended memory and then on to disk.

```
#include <alloc.h>
#include <heap.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>

void Message (int t)
{
  printf("\nSwapping...");
}

void main (void)
{
  MHNDL hString;
  char *p;

  /* set up pointer to function that */
  /* will be called during swap */
   swp_msg = Message;

/* allocate 2 KB from virtual memory */
hString = vmmalloc(2048);
if(hString == NULL)
{
  printf("\nFailed to alloc memory");
}
else
{
  /* Assign a pointer to the handle */
  /*  in order to use the memory */
  p = vmderef(hString, NO_LOCK);
  strcpy(p, "Hello World");
  printf("\n%s", p);
  /* Free virtual memory */
  vmfree(hString);
}
exit(0);
}
```

Figure 2 - MoreHeap Virtual Memory

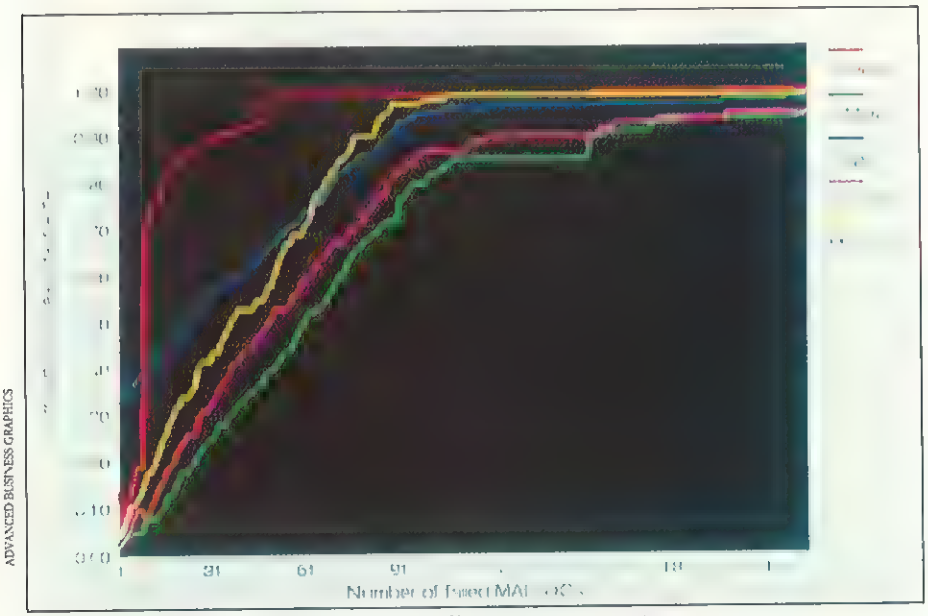


Figure 3 - Malloc Fragmentation

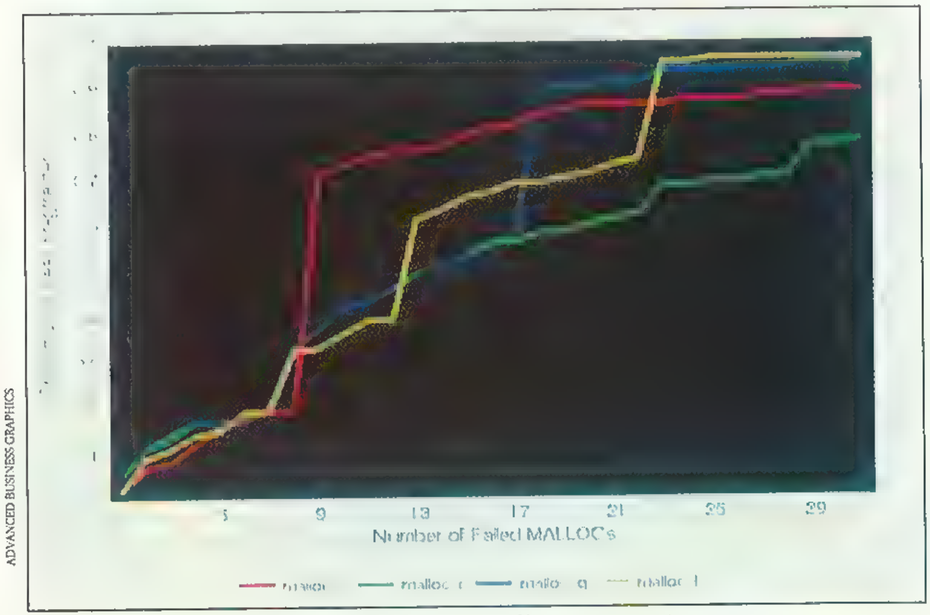


Figure 4 - C-Heap Malloc Comparison

The virtual memory manager provides three main types of memory block and uses handles to access blocks of virtual memory. In order to read or write to a block, the handle must be dereferenced to obtain a pointer to a physical block of memory (see Figure 2). Once it has been dereferenced, a memory block is locked and it will not be swapped to disk until it has been unlocked. This guarantees that a dereferenced block will always be in physical memory and so pointers to the block will be valid. Next there are discardable memory blocks which behave like normal virtual memory until you run out of physical memory. When this occurs, discardable blocks that have not been locked are thrown away without getting swapped to disk. As long as the handle to a discardable memory block is valid, the memory block is present in physical memory. This type of memory can be used to store files that do not get modified. Finally there are the swappable memory blocks which must be copied to disk when free physical memory is low. MoreHeap also provides relocatable memory blocks which are never swapped to disk. These blocks can be moved in order to defragment the heap. This occurs when the programmer invokes the mhcompactheap routine.

Unlike Dolphin, debugging is quite limited. It provides a heap log and it is possible to check the integrity of the heap automatically after calls to malloc, calloc and free. When an error occurs, MoreHeap displays the line number and source module where the heap function was called.

XMem

Unlike the other three libraries, XMem provides a consistent interface for accessing memory, whether it is conventional memory, EMS, XMS or virtual memory. It uses handles to access all memory and therefore it cannot directly be used as a replacement for the memory management routines in the C run-time library. However, with a little insight, and a single #define, XMem can be made to emulate malloc.

```
#define malloc(n)              \
  xLock(xAlloc(n, XH_LOCKED \
            0))
```

In the above macro, xAlloc returns a handle to an n-byte block of memory which

Nu-Mega
TECHNOLOGIES INC

Get Inside WINDOWS!

MICROSOFT WINDOWS
Version 3.0 Compatible Product

Debug Windows at the systems level!

Soft-ICE/W takes you inside Windows! Debug and explore with power and flexibility not found in any other Windows debugger! Soft-ICE/W allows you to debug at the systems or applications level or simply learn the inner workings of Windows.

- Debug VxD's, drivers and interrupt routines at source level
- Debug interactions between DOS T&SR's and Windows Apps
- Debug programs in DOS boxes
- Display valuable system information (from the total memory occupied by a Windows application, to the complex internal structures of Windows)

Soft-ICE/W uses the 386/486 architecture to provide break point capabilities that normally require external hardware. Nu-Mega, which pioneered this technology with the introduction of its award winning Soft/ICE for DOS, now gives Windows programmers the same debugging power... *and still at a software price.*

Own the debugger that combines the best "view" of Windows internals with the most powerful break points of any software debugger.

Soft-ICE/W . . . Only £259.00 +VAT

CodeView for Windows users: see what you're debugging without flash. **CV/1 version 2.0** runs CodeView in a graphics window while viewing your application screen. Runs on <u>any</u> display that supports Windows.

CV/1 . . . Only £95.00 +VAT

WHAT THE EXPERTS ARE SAYING

"Soft-ICE for Windows is great! It helped me find, in fifteen minutes, a killer bug in a Windows virtual device driver that had eluded two people for several months. I can't see doing Windows development of any kind -- whether writing Windows applications, device drivers, or even DOS programs that have to run under Windows -- without it. In addition to being great for finding bugs, Soft-ICE for Windows has been essential for my work on a forthcoming book: on Undocumented Windows. Soft-ICE for Windows goes anywhere and does everything, so it's essential for anyone who wants to poke around inside Windows Enhanced mode. DOS programmers will find it a perfect way to learn how the Windows DOS extender and DPMI server work, and how Windows interacts with DOS. Windows Enhanced mode is the hacker's paradise of the 90s, and Soft-ICE for Windows is the tool that every serious Windows or DOS hacker will need. Nu-Mega has done a brilliant job!"

Andrew Schulman
Software Engineer, Phar Lap Software
Editor, Undocumented DOS
Coauthor, Undocumented Windows (forthcoming)

Tel: (071) 833 1022
Fax: (071) 837 6411

3-5 Cynthia Street,
London N1 9JF

Now in the UK
System Science

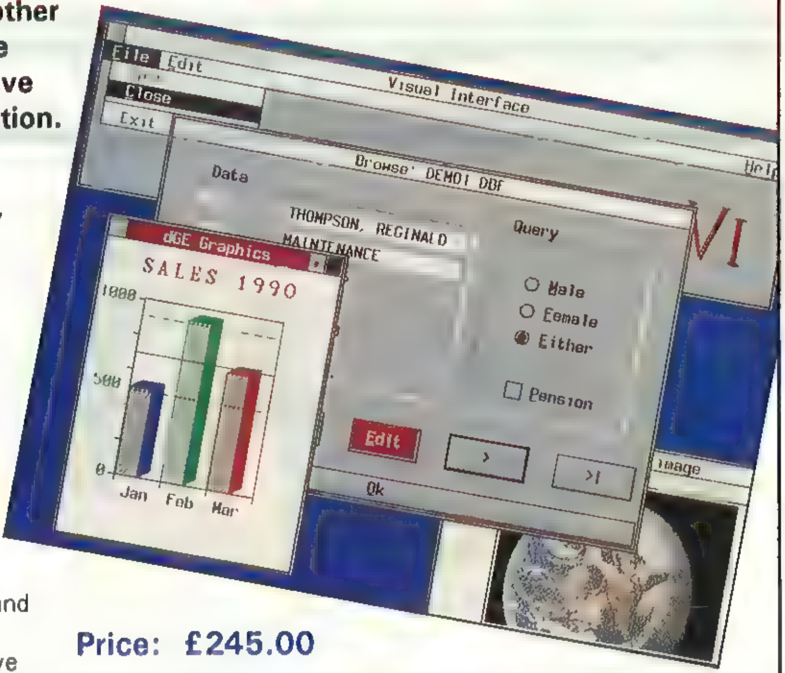
RISK = NULL
30 DAY
MONEY-BACK GUARANTEE

MICROSOFT WINDOWS IS A REGISTERED TRADEMARK OF MICROSOFT CORP. Soft-ICE/W AND CV/1 ARE TRADEMARKS OF NU-MEGA TECHNOLOGIES, INC.

CIRCLE NO. 834

VISUAL INTERFACE GUI Library for Clipper

We led the way with dGE. This is the other indispensable library for Clipper. For the first time your Clipper programs can have the look and feel of a Windows application.



Visual Interface has it all....

- Multiple windows that can be moved, sized, maximized and minimized
- A complete range of interactive controls
- An integrated event handler
- Simple to program
- Fully object-orientated
- Runs dGE graphics in it's windows

Your applications will never be the same....

- Create graphical dialog boxes, data-entry forms and context-sensitive help screens
- Select files from scrolling list boxes
- Import bit-mapped images
- Create queries using icons, radio buttons, and check boxes
- Run several windows concurrently and move between them with a keystroke or mouse-click

VI comes with a 250 page manual including over 100 pages of tutorial with examples.

Requires:
Clipper 5.01
DOS 3.x or higher
640K RAM
EGA or VGA

Price: £245.00

From: Bits Per Second Limited
14 Regent Hill, Brighton, E.Sussex BN1 3ED
Tel. 0273 727119 Fax 0273 731925

The VI library contains elements of Class(y) and SuperClass issued under licence. You do not need to buy these products to use VI with Clipper 5.01
Class(y) (c) Anton van Straaten
SuperClass (c) Chydale Software
dGE, Visual Interface (c) Bits Per Second Ltd.

CIRCLE NO. 835

LOOK at all OPTIONS.

Software Protection - but which TYPE ?

- **EVERKEY II** — *The Hard ware Option?*
  SINGLE WIRE ZERO LOAD - Total Compatibility. Not available in ANY other Key product.
- **EVERLOCK** — *The Soft ware Option?*
  Software-Only Copy Protection; economical & SECURE.
- **EVERTRAK** — *Non Copy-Protection Option?*
  Non Copy-Protection - Anti Hacker software protection.

AZ TECH SOFTWARE
PRINTER
AZ
EVERKEY
COMPUTER

CALL FOR A FREE WORKING DEMO DISKETTE

Other Services include:- Disk Duplication & supply - Label Production & supply - Packaging.

USER FRIENDLY
MicroSystems
22A, Bartleet Road,
Washford Industrial Estate,
Redditch, Worcestershire,
B98 0DG. England.
Tel: 0527 510 105
Fax: 0527 514 229

Sole Distributor for Az-Tech Software,inc - UK, Ireland & Spain.

......THE ONLY OPTION.

CIRCLE NO. 836

has been locked so that it doesn't get swapped to disk. The `xLock` function returns a pointer to the block.

XMem must first be initialised before calling routines in the library. During initialisation you have to specify the maximum number of handles that you intend to use, the number of swap pages and a list of swapping strategies. The number of swap pages determines the size of the swap area and the swapping strategies tell XMem where to swap blocks of memory when it runs out of conventional memory. For instance, the follow code extract will inform XMem to allocate 256 handles and use conventional memory, then XMS before swapping to disk. The size of the swap area is 512 KB (32 * 16 KB swap page size).

```
xStartup(256, 32,
   S_EXTENDED,
   S_DISK,
   S_ENDLIST);
```

One of the problems I encountered with XMem is that you have to create a swap file before XMem can swap to disk (ie it doesn't create one for you). This file must be large enough to accommodate the entire swap area defined during initialisation. So the code above would require a 512 KB swap file to exist on your hard disk.

Like MoreHeap, blocks of memory can be locked, discardable or swappable. The heap is compacted by XMem automatically whenever a library call is invoked, although a programmer can call the `xCompress` routine manually. During compaction, swappable blocks are rearranged and discardable blocks are deleted in order to increase the amount of free heap space.

A nice feature of XMem is that it provides a set of functions which emulate the memory management routines in Windows, OS/2 and the Macintosh. This provides a neat way of easing the porting of text-only applications from DOS to these platforms. The Windows emulation library provides seven routines for global allocation only (eg `GlobalAlloc`). There are also 12 Macintosh-style and five OS/2 style library calls.

Glossary

Compaction - A process of moving memory blocks around in order to increase the size of the **Free Memory Pool** and thus reduce **Fragmentation**.

Conventional Memory - This is the block of memory from address 0 to address 655,359 which can be accessed from DOS directly. Special device drivers are needed to get at memory above the 640 KB driver. *See* Upper Memory Area.

EMS - The Expanded Memory Specification provides a mechanism for addressing more than 640 KB of **Conventional Memory**. When a program needs to access expanded memory, the expanded memory manager maps a 16 KB page from expanded memory onto a page frame in the **Upper Memory Area**.

Fragmentation - Occurs when blocks of memory are freed without being returned to the **Free Memory Pool**. Fragmentation usually occurs when a free block is too small for reallocation by the **Heap Manager**. *See* Free Memory List.

Free Memory List - A linked list of memory blocks that have been deallocated using `free`.

Free Memory Pool - A contiguous block of unallocated memory which the **Heap Manager** dips into when the **Free Memory List** can't provide a block large enough to satisfy a given `malloc` request.

Heap Manager - A set of routines which coördinates the actions of `malloc` and `free` in a program. Memory is allocated from the **Free Memory Pool** and freed into the **Free Memory List**.

HMA - (High Memory Area) The first 64 KB above 1 MB which DOS can access using the HIMEM.SYS driver.

UMA - (Upper Memory Area) the 384 KB of memory above the 640 KB of conventional memory.

Virtual Memory - A way to access masses of memory without having to spend a fortune on installing extra RAM. The Virtual Memory Manager uses a hard disk to swap blocks of memory in and out of physical memory.

XMS - The eXtended Memory Specification enables a DOS application to access memory above 1 MB on a 286/386 machine. However, this memory is only accessible in protected-mode so an Extended Memory Manager like HIMEM.SYS is required.

XMem doesn't have a heap log or a mechanism for examining the contents of the heap (ie a Heap walking routine). It does provide six error codes and these can be captured in a error handling function (XMem uses the global variable `xerr_handler` to point to an error handling function supplied by the programmer).

C-Heap

C-Heap is a superset of the C run-time library, offering the programmer over 500 memory management routines. With several versions of each memory management routine, it is easy to get overcome by the sheer bulk of the library. However help is at hand. Documentation is superb. There's a user manual and a 1006 page functional reference. Putting the hefty functional reference to one side, the user manual describes how to make best use of the C-Heap library. It looks at the way C-Heap manages memory and it outlines the main functions in the library. There are also sections on fragmentation and garbage collection.

As is the case with MoreHeap, the C-Heap library can be used without having to change any application source code. C-Heap splits the Heap into various sections and provides several routines for managing these separate areas.

`Malloc` and its derivatives are allocated from the bottom of the heap - ie the heap grows upwards. This is called the **bottomfarheap** and is the familiar heap model found in the run-time library. C-Heap also offers routines which allocate from the top of the heap, making the heap grow down towards lower memory. The C-Heap user manual calls this heap the **topfarheap** and provides routines like `malloctop` and `calloctop` for allocating memory from it. This provides a fast way of freeing several blocks simultaneously.

Another group of functions (the TopNode routines) let you allocate directly from the free memory pool without scanning through the free memory list. This has the advantage of a significant increase in speed

The Heap Manager Benchmark

All benchmarks were performed on a 40 MHz 80386 with 4 MB of RAM running MS-DOS 5.0. MEM reported 490.5 KB of free conventional memory and 13.0 KB of free upper memory at the DOS prompt.

The machine was configured with a 2 MB RAM disk in extended memory. The times for `malloc` and `calloc` were obtained by allocating random blocks of memory until there was no memory left and then freeing all the blocks. This process was repeated several times to obtain an accurate result.

In addition to these routines, I decided to time the heapwalking functions since heapwalking forms an important part of heap compaction. The speed of this was obtained by timing how long it took the heapwalker to visit every node on the heap. The RAM disk was used to provide a consistent swap area for the virtual memory benchmarks.

In the fragmentation test (or Figures 3 & 4) random blocks of memory were allocated until allocation failed. After freeing all the blocks, the degree of fragmentation was found by calculating the proportion of unusable memory (memory which was still allocated after all blocks had been freed) to the total amount of memory available on the heap.

at the cost of greater fragmentation (since the free memory pool shrinks).

To get around fragmentation, C-Heap provides the programmer with `mallocdole` which enables fixed size blocks to be allocated from a contiguous block of memory. `mallocdole` must be initialised before it can be used. During initialisation, `mallocdole` has to be told the size of the elements in the array. The first time `mallocdole` is called, it allocates space for an array of 32 elements from the heap and then returns one block from the array. With further calls to `mallocdole` more memory is spooned out until the array becomes exhausted. When this occurs, another contiguous block of 32 elements is allocated from the heap.

To reduce fragmentation even further, C-Heap can be told to increase the minimum size of a heap node. If this is increased from 1 byte to 60 then, when a block of 20 bytes is allocated, the entire 60 byte node is used causing 40 bytes of memory to be wasted. However when the 20 byte block is then freed, a node containing 60 bytes is added to the free node list for reallocation. This reduces the number of unusable free nodes in the heap and thus reduces fragmentation.

C-Heap also provides several routines which return information about the heap and supports XMS and virtual memory.

Conclusion

So what can be deduced from the benchmarks? Well, if you're looking for the fastest `malloc or calloc` then you should consider C-Heap or XMem, although Borland's own library has performed remarkably well and its HeapWalker left the other libraries on the starting line. Looking at fragmentation we can see that XMem is the winner with the least, but this time, Borland has taken second place. I can see that there's a big question mark hanging over your decision to opt for one of these third-party libraries - after all, Borland seems to be doing just fine.

However, there are some things that Borland doesn't offer. For a start it gives precious little in the way of debugging. Dolphin provides the most comprehensive heap debugging in the bunch but it lacks performance. Using a couple of `#defines` you can use Dolphin without having to alter any source code. This could be useful during the early stages of development for debugging purposes.

The ability to acces EMS/XMS/Virtual Memory is arguably the single most important feature that Borland lacks. (Users of the Zortech C++ compiler have a method for accessing EMS, manipulated using handles.) MoreHeap, C-Heap and XMem all provide routines for accessing EMS/XMS/Virtual memory, although Dolphin doesn't support this feature. In this category, MoreHeap is the fastest.

It is always difficult to pick a favourite. Dolphin provides the best debugging, MoreHeap is easy to use and C-Heap provides you with almost every conceivable memory management routine. But for its consistent interface and support of Windows, OS/2 and Mac memory management, I would choose XMem.

EXE

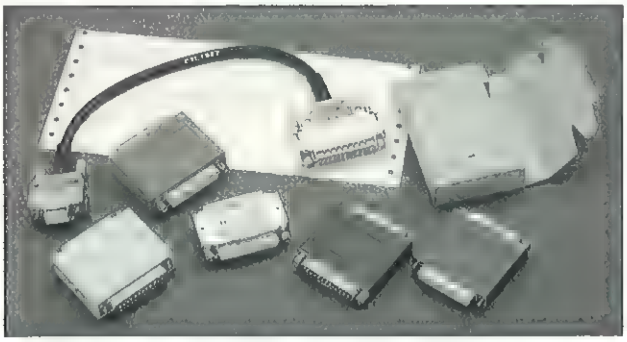
C-Heap is produced by Library Technologies on 0101 608 2744224. Dolphin can be contacted on 0101 510 4643009. MoreHeap is produced by SeaBreeze Software Systems on 0101 609 9246793. XMem is distributed by Great Western Instruments on 0761 452116.

SOFTWARE SECURITY MODULES

Hardware devices (dongles) are a recognised and proven means of protecting software from unauthorised use and piracy.

Our range of devices offers some of the most robust and troublefree solutions around. All units are cascadeable and can be uniquely coded for each customer. Features include:

- *PC/non-PC.*
- *RS232/Printer.*
- *Internal memory (some units)*
- *Software drivers supplied.*
- *Minimum 2 year guarantee.*
- *From £14 to £50*



CTL

Control Telemetry of London
11 Canfield Place, London NW 3BT
Tel: 071 328 1155 Fax: 071 328 9149

CIRCLE NO. 838

EIFFEL:
NOW AVAILABLE FOR MS-DOS

We are pleased to announce the release of Eiffel/S, the Eiffel compiler for MS-DOS.

Eiffel enables you to produce efficient, well-documented, reusable software, taking full advantage of the benefits offered by object-oriented technology.

Eiffel/S, written in version 3 of the Eiffel language, is fully approved by the International Eiffel Consortium. Features inclde:

- Extensive libraries: basic classes, I/O classes, persistence classes, data structure classes
- Production of optimized C code
- Fast syntactical analyzer
- Platform independence
- Eiffel/S is available for MS-DOS, OS/2, UNIX.

For further information contact:

Applied Logic Developments
9 Princeton Court
55 Felsham Road
London SW15 1AZ
Tel: 081 780 1088
Fax: 081 780 1941

Applied Logic

CIRCLE NO. 839

LIANT C-scape® 3.2

The latest release of C-scape – the best selling character/graphics based screen interface C library in the UK. Runtime royalty free Includes Look and Feel™ Screen Designer and Source Code. Available for DOS, most DOS Extenders, UNIX, X Windows and VMS.

DOS – **£370** (incl source and LnF)
UNIX – **£1065** (incl source and LnF)

OTHER LIANT PRODUCTS

LPI C	– ANSI standard C for UNIX	**£660**
	– ANSI standard C for DOS	**£695**
LPI C++	– for UNIX	**£975**
LPI COBOL	– for UNIX	**£950**
	– for DOS	**£875**
LPI FORTRAN	– for UNIX	**£975**
	– for DOS	**£875**
LPI PASCAL	– for UNIX	**£950**
LPI – PL/1	– for UNIX	**£2200**
	– for DOS	**£1095**

Heron Software C-scape Extension Library

New version 2
Enhanced set of functions for C-scape 3
Includes Source Code.....

	Price
DOS	**£150**
Unix	**£250**

Phar Lap 386|DOS Extender™

Beat the 640K DOS limit with a flat 32-bit address space on 386 processors. Voted best of 1989. Supports a wide range of 32-bit languages. .....SDK **£330**

Phar Lap 286|DOS Extender™

C beyond 640K with your Borland C++, Microsoft C or MS Fortran compiler on any 80286, 386 or 486 platform....SDK **£330**

Raima Data Manager™ Release 3.21

Complete family of C library dbms functions and database management utilities for DOS, Windows, OS/2, Macintosh, UNIX, QNX, VMS. Source code available – runtime royalty free.
Single-user DOS, OS/2 or Windows DBMS object libraries. ... **From £460**

Ask for details of other options and multi-user pricing.

Magna*Charter* II

Enhanced version of the much acclaimed Magna*Charter* charting package for Windows 3, 99x99 cells with wide range of printer support. Complete set of symbols – or add your own . **£160**

Still available, Magna*Charter* for DOS (includes runtime Windows 2.)..... **£118**

SNIPPETS™ INFORMATION MANAGER

NEW – Intelligent free text organiser and text retrieval system. Complete personal information manager or corporate information system.
DOS S/U **£175** DOS M/U **£625**

PERSONAL ACCOUNTANT

The only complete personal finance package – for DOS **£149.95**
NEW – Tax Reckoner for UK tax payers – for DOS **£119.95**
(Personal Accountant prices include VAT).

All Systemstar products are distributed and supported by Systemstar. Prices are exclusive of VAT (except Personal Accountant) and subject to change. Orders and enquiries to:

SYSTEMSTAR LIMITED

1-3 Parliament Square, Hertford, SG14 1EX
Telephone: (0992) 500919 Facsimile: (0992) 554261

CIRCLE NO. 840

3D Computer Graphics - The Z-Buffer Algorithm

The Z-Buffer algorithm is conceptually very simple - yet is able to handle complex scenes almost as readily as simple ones, as Graeme Webster explains.

Previous articles in this series have presented a basic graphics library which can be used to exploit the high resolution 256 colour modes of Super-VGA graphics cards and the essential algebra and routines for 3D manipulations and perspective transformations. The October 1991 issue of *.EXE* described the 'Warnock algorithm', one of the all-time classics of 3D computer graphics. The Warnock algorithm is very powerful and very general in being able to deal equally well with hidden line or hidden surface removal. It is essentially device independent and can generate images to any desired resolution, making it especially useful for images plotted on paper. However, the advent of inexpensive, reasonably high resolution colour displays makes other methods, which exploit the hardware more directly, attractive as alternatives. The Z-Buffer algorithm is one such.

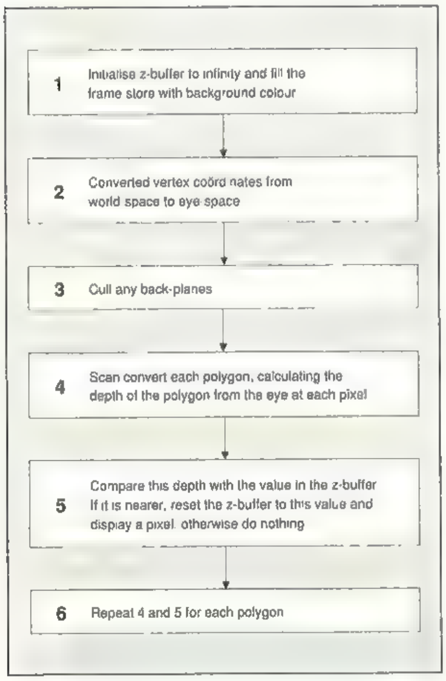

1 Initialise z-buffer to infinity and fill the frame store with background colour
2 Converted vertex coörd nates from world space to eye space
3 Cull any back-planes
4 Scan convert each polygon, calculating the depth of the polygon from the eye at each pixel
5 Compare this depth with the value in the z-buffer If it is nearer, reset the z-buffer to this value and dispiay a pixel otherwise do nothing
6 Repeat 4 and 5 for each polygon

Figure 1 - The Z-Buffer algorithm in outline

Despite - or perhaps because of - its simplicity, it has stood the test of time. As we will see below, there are a number of major problems with the algorithm as one pushes on towards ever greater realism. Nevertheless, it is easy to implement in hardware and, as a result, is to be found both in graphics workstations and in advanced microprocessor chips such as the Intel 80860.

The Algorithm

As with Warnock, the Z-Buffer algorithm is really a generic title for a group of related techniques which all exploit the same basic idea. It was originally proposed by Edwin Catmull, another of the University of Utah computer graphics giants.

The idea is very simple. It starts from the premise that figuring out which bits of a surface can be seen and which bits are hidden behind others is hard work. It becomes especially tricky if surfaces can inter-penetrate or can overlap cyclically, eg surface *A* covers part of surface *B* which covers part of *C* but *C* covers part of *A*: so lets not do it!

What to do instead? Well, the attributes, intensity, colour etc, of every pixel on the screen are stored in a frame buffer. For the Super-VGA cards, which are the primary concern of this series of articles, there are 8 bits per pixel. In its simplest form, the *z*-buffer is a separate buffer used to store the depth or z-coördinate of every visible pixel in image space.

The first step in using the buffer is to initialise its contents to values which signify 'infinitely far away' and, if desired, fill the frame store with a background colour or scene. Next, the coördinates of the vertices of the polygons of the objects making up the scene to be displayed are converted from world-space to eye-space *(see .EXE, May 1991)*. At this point any back-planes can be culled *(see .EXE, October 1991)*. A polygon scan conversion *(see .EXE, March 1991)* is now carried out with a special feature that at every pixel which is a candidate for display, the depth of the polygon from the eye is found. This depth is compared with the value in the *z*-buffer corresponding to the pixel. If the pixel depth is less than the stored value, the point lies in front of anything which has already been displayed, so the pixel is reset to the colour of the current polygon and the value in the *z*-buffer changed to be the new depth. If the depth corresponding to the pixel is greater than that already in the *z*-buffer then the new pixel is invisible (it lies behind other material that has already been processed) and nothing else needs to be done. Figure 1 shows the process diagrammatically.

The simplicity of the algorithm is its great advantage. It deals with the problem of hidden planes or curved surfaces, partial and cyclical overlapping and with interpenetration with complete indifference. As the size of the image is fixed the processing time increases only about linearly with complexity. Polygons can be written to the frame store and *z*-buffer in an arbitrary order, eliminating the need for any kind of depth sort. Some minor improvements in performance can be had by dealing with nearer polygons first, while back-plane culling saves setting values which are certain to be overwritten. Finally, adding texture and shading is straightforward.

SOLUTION SYSTEMS PROGRAMMING TOOLS GIVE YOU LEVERAGE.

Put the physics of programming to work for you. ¶ Right out of the box, together or separately, Solution Systems products give you leverage that will send your productivity soaring. It all begins with BRIEF. Our programmer's editor is the favorite of programmers around the world, renowned for its speed, configurability, and ease of use. ¶ dBRIEF for X-Base and Paradox are award-winning add-ons that custom tailor BRIEF for DBMS development. BRIEFor C++ adds a C++ browser to BRIEF for those of you with an object orientation. ¶ Sourcerer's Apprentice provides version control that's seamlessly integrated with BRIEF. It's never been easier to keep track of development on individual PCs and Novell or UNIX-served networks. ¶ And then there's C-Worthy. The User Interface Development System. C-Worthy gives you the menus, screens, colors, and screen designer you need to create great-looking applications at warp speed. ¶ When you're ready to wrap it all up, CHARGE, an interactive code profiler with direct access to BRIEF, will give you new insight into your code's performance. ¶ Raise your productivity and creativity to new heights. Apply the physics of programming to your applications today.

AUTHORISED DEALERS:-
Absolute Computing 0923 56043 • **Bonsai** 071-631 5454 • **Computer Solutions** 0932 352744 • **Computacentre** 0923 56040 • **Computerland** 0256 816611 • **Corporate Computers** 0245 450039 • **Grey Matter** 0364 53499 • **International Software** 081-479 0049 • **Metrocom** 081-740 6666 • **P&P Corporate** 0706 217744 • **Roundhill** 0672 84535 • **Software Box** 0904 783036 • **Software Construction Co.** 0763 244114 • **Software Corporation** 0904 691391 • **System Science** 071-833 1022 • **Technomatic** 081-205 9558 • **Technology Plc** 0925 818220 • **Worldwide Computers** 081-543 2211.

Solutionssystems

1 The Maltings, Green Drift,
Royston, Hertfordshire SG8 5DB
Telephone: 0763 244141
Facsimile: 0763 244025

CIRCLE NO. 841

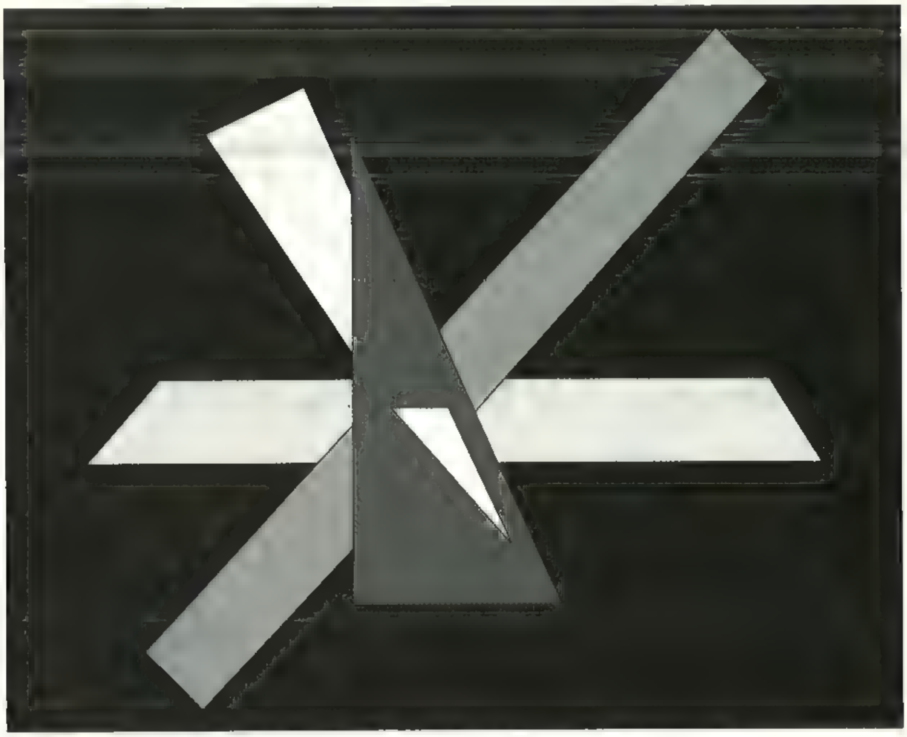


Figure 3 - Line drawing of program output

The method is not without its disadvantages. For a long time the principle one was the amount of memory required. Depth must be held to higher precision than lateral information. In the past, 20 bits per pixel was typical, but a floating point buffer with 32 bit floats is better. For a 1024x768 pixel image we are, therefore, talking about 3 MB for the *z*-buffer. If this much memory is not available in RAM, it is possible to hold it on hard disk or to segment the image into a number of rectangles or strips. Subdivision generally leads to slower processing because each polygon tends to be handled several times even if they are sorted by area in some way.

Another disadvantage is the difficulty of implementing the transparency and translucency effects which increasingly are demanded in the quest for photorealism, because of the arbitrary order in which pixels are written into the buffer. Anti-aliasing by sub-pixel averaging is easy to implement but expensive on memory. Just doubling the image spatial resolution while creating the image, and then performing anti-aliasing by simple averaging, puts up the buffer requirement to 12 MB.

Although transparency and the like is difficult, shadows are quite easy - provided that memory is no object. A separate shadow *z*-buffer is associated with each light source. The production of an image is accomplished in two stages. The first processes the scene, taking the light source as the eye-point. The shadow *z*-buffer is used to store depth information only, no pixels being rendered at this stage. The second step is a normal *z*-buffer rendering with an additional twist. If a pixel is visible, a coördinate transformation is used to map the coördinates of the point in the 3D screen space with the eye point as origin, to screen space with the light source as origin. If the depth in the shadow *z*-buffer is less than the transformed depth of the point under consideration, then the latter is in shadow; otherwise it is fully lit. As with *z*-buffers generally, the attraction of this technique is its total indifference to complexity which can floor other methods. The down side is again the vast amount of memory required for multiple light sources or serious loss of performance if the same shadow buffer is reused.

Scan line Z-Buffering

If memory really is a problem you can keep making the slices thinner and thinner until, in the limit, you are left with a single scan line! With full screen buffering the natural approach is to deal with one polygon completely before going on to the next polygon. In contrast, a scan line *z*-buffer algorithm will work with one scan line at a time, testing each polygon against it, before moving on to the next line. An efficient way must be devised for handling the polygons or an enormous amount of wasted work will be done. At very least the upper and lower limiting scan lines need to be pre-calculated for each polygon so that ones not cut by the current scan line can be quickly identified and skipped.

Efforts were made to try to improve the efficiency of this technique by introducing the concept of 'spans'. Spans are consecutive groups of pixels along a scan line that are either empty, so that the background shows through, are covered by a single polygon or are covered by several polygons. Spanning algorithms seek to work out where the edges of such spans are and then use that knowledge to facilitate the calculation of depths.

In practice, spanning scan line algorithms only show advantages for scenes of low complexity and, with the wide availability of lots of low cost memory, scanning methods generally lose out to the full screen *z*-buffer technique.

The Code

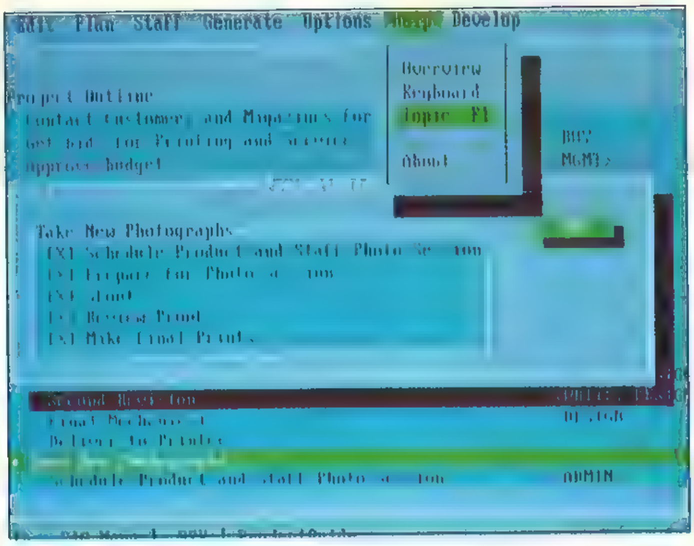
In this article I again present only the simplest of possible implementations, stripped of complications which tend to obscure what is going on. Ideally we should use a single 1024x768 *z*-buffer for the image. With the new 32-bit compilers working in protected mode (see future article) this is indeed what one would do. For the moment we will stick to standard Microsoft C under real mode MS-DOS, a corollary of which is that one can afford only about

```
Number of polygons 4
Polygon 0:
Colour 7
Number of vertices 3
x y z of vertices
0 -50 -100
0 100 -100
0 -50  100
Polygon 1:
Colour 56
Number of vertices 4
x y z of vertices
-10 -200 -20
-10  200 -20
-10  180  20
-10 -180  20
Polygon 2:
Colour 192
Number of vertices 4
x y z of vertices
-5 -165 -135
-5 -135 -165
-5  165  135
-1  135  165
Polygon 3:
Colour 63
Number of vertices 3
x y z of vertices
-30 -120  80
-40  -80 120
 30   50 -50
```

Figure 2 - Sample data for program

A PLANNING BREAKTHROUGH FOR EVERY ORGANISATION

- Speeds planning by up to 90%
- Improves planning quality at all user levels
- Eliminates errors, omissions and time-consuming repetition
- Reduces keystrokes by a factor of more than 20
- Standardises planning methodologies throughout large organisations
- Facilitates broad-scale use of project management techniques



Project Outline
Contact Customers and Magazines for
Get bid for Printing and services
Approve budget

Take New Photographs
[X] Schedule Product and Staff Photo Session
[X] Prepare for Photo Session
[X] Shoot
[ ] Review Proof
[X] Make Final Prints

Second Revision
Final Mechanical
Deliver to Printer
Schedule Product and Staff Photo Session ADMIN

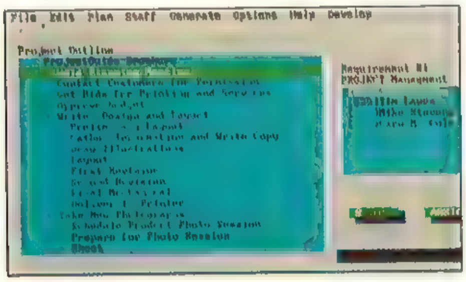
ProjectGuide is available in two editions: Runtime for end-users, and the Professional Edition for Guide development.

The Professional Edition provides an integrated, interactive Guide development environment that requires only limited programming expertise.

Features include automated error checking, and reverse engineering tools that allow you to import tried and tested plans from existing project management software.

ProjectGuide
The Expert Project Plan Generator

ProjectGuide is the world's first general-purpose project plan generator, designed to allow both experts and non-expert users to produce comprehensive, well-considered plans at the first try.



Project Outline
ProjectGuide Brochure
Contact Customers for Permission
Get Bids for Printing and Services
Approve Budget
Write, Design and Layout
Prepare for Photo Session
Shoot

Requirement #1
PROJECT Management

Working with easy-to-use question and answer dialogues and checklists, ProjectGuide creates individual project plans - complete with task lists, dependencies, estimates and resourcing - that can be loaded straight into the most popular project management systems.

It uses a unique requirement-based planning approach in which users enter requirements and ProjectGuide proposes tasks - a much simpler process than conventional project management systems, where users have to enter full task details themselves.

This dramatically speeds the planning process, reduces the potential for errors and enhances the quality of any project plan.

Less typing - more planning

ProjectGuide automatically defines a logical workflow, so you don't spend hours 'getting the connections right'. The easy outline-style dialogue operation can reduce key-stroking by a factor of more than 20.

In fact, ProjectGuide has been shown to cut planning time by up to 90%.

Capture and access valuable planning knowledge

Planning knowledge is stored in interactive scripts called 'Guides', with each Guide containing expertise on a particular class of project.

These Guides enable every organisation to capture and disseminate planning knowledge, experience and methodologies and estimating techniques. Then any user throughout the organisation can simply select the appropriate Guide and carry out their own planning, using a consistent methodology.

Better planning throughout your organisation

ProjectGuide maximises the return on investment in project management software by making the system readily available to the broadest possible range of users.

Fundamentally simple in operation, it can at any level, from department and project managers to line managers and project planners.

With appropriate Guides it will simplify virtually all planning applications - from standardising new product development to formalising client proposals and tendering procedures.

Generates complete plans for all major project management software:

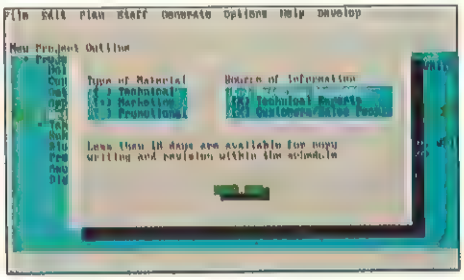
- CA SuperProject V2 DOS & Windows
- Project Manager Workbench
- Microsoft Project for Windows
- Time Line • InstaPlan • PLUS links to Time$heet professional and other management software.

(ProjectGuide for DOS with optional mouse support is available now. Windows version available May 92.)

CIRCLE NO. 842

Full training and consultancy

A full range of training and consultancy is available for both Runtime end-users and Guide developers, and for in-house and external Guide development.



New Project Outline

Type of Material
( ) Technical
(•) Marketing
( ) Promotional

Source of Information
[X] Technical Reports
[X] Customers/Sales People

Less than 10 days are available for copy writing and revision within the schedule

For full details send for your free information pack by posting the coupon below or faxing it on 081-423 8992. Alternatively, call us direct on 081-423 8855.

.EXE 4/92

PLEASE SEND ME INFORMATION ON:

□ ProjectGuide

Other Deepak Sareen Associates products:

□ Time$heet PROFESSIONAL (time and expense tracking, links with PM systems)

□ InstaPlan (best-selling project management software)

□ TimeSlips (time and expense recording and billing)

□ I would be interested in attending a seminar on some or all of these products

Name ______

Position ______

Company ______

Address ______

______

Postcode ______ Tel No. ______

No. of PCs in your organisation/department ______

Are PCs networked? YES □ NO □

Do you currently use project management software? YES □ NO □

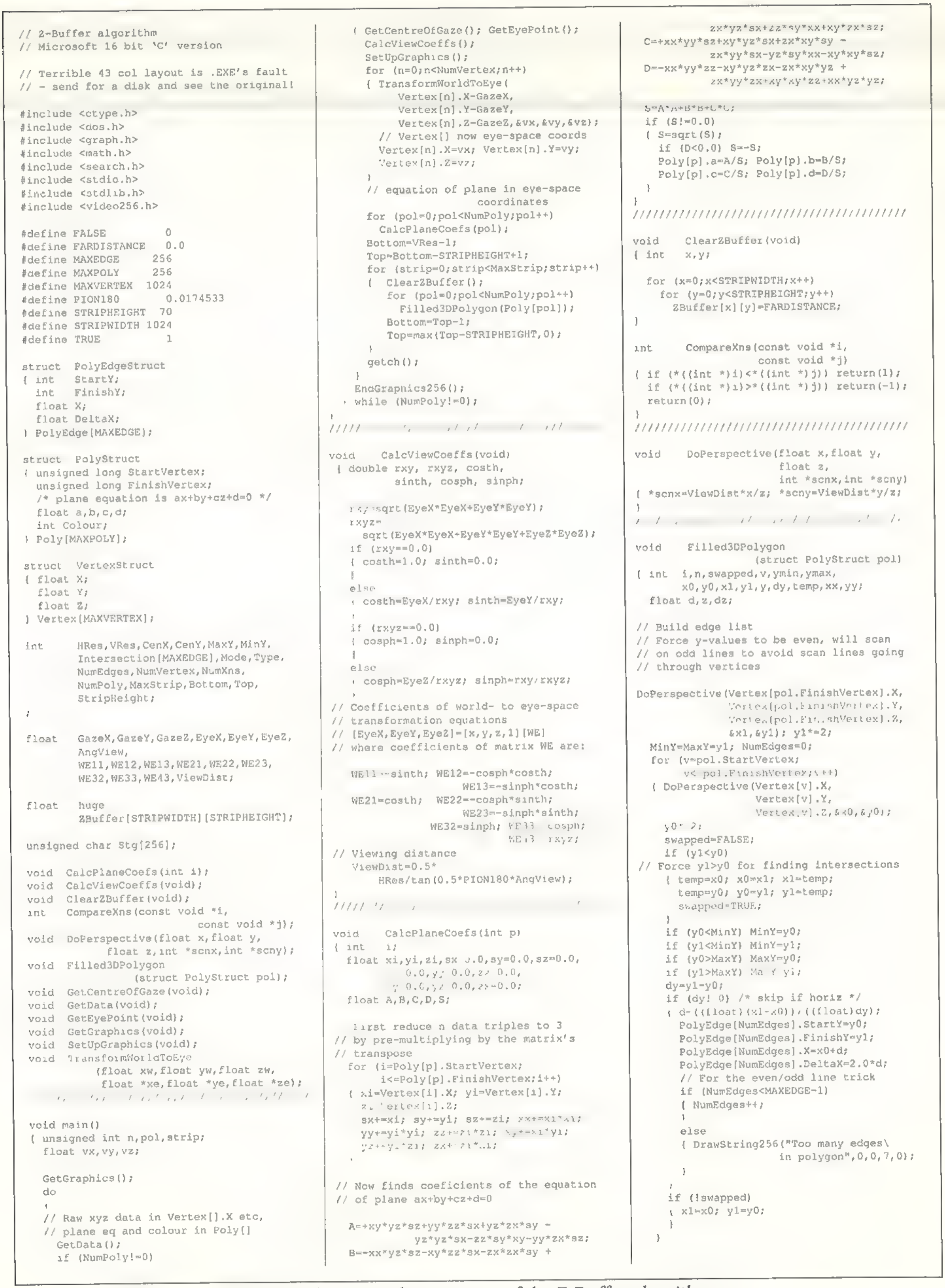
If yes, please state which ______

Do you currently use time/expense tracking software? YES □ NO □

If yes, please state which ______

DEEPAK SAREEN ASSOCIATES

Return to:
Deepak Sareen Associates,
FREEPOST, Bydell House,
Sudbury Hill, Harrow-on-the-Hill
Middlesex HA1 3BR



```
// Z-Buffer algorithm
// Microsoft 16 bit 'C' version

// Terrible 43 col layout is .EXE's fault
// - send for a disk and see the original!

#include <ctype.h>
#include <dos.h>
#include <graph.h>
#include <math.h>
#include <search.h>
#include <stdio.h>
#include <stdlib.h>
#include <video256.h>

#define FALSE          0
#define FARDISTANCE    0.0
#define MAXEDGE      256
#define MAXPOLY      256
#define MAXVERTEX   1024
#define PION180        0.0174533
#define STRIPHEIGHT   70
#define STRIPWIDTH  1024
#define TRUE           1

struct  PolyEdgeStruct
{ int   StartY;
  int   FinishY;
  float X;
  float DeltaX;
} PolyEdge[MAXEDGE];

struct  PolyStruct
{ unsigned long StartVertex;
  unsigned long FinishVertex;
  /* plane equation is ax+by+cz+d=0 */
  float a,b,c,d;
  int Colour;
} Poly[MAXPOLY];

struct  VertexStruct
{ float X;
  float Y;
  float Z;
} Vertex[MAXVERTEX];

int       HRes,VRes,CenX,CenY,MaxY,MinY,
          Intersection[MAXEDGE],Mode,Type,
          NumEdges,NumVertex,NumXns,
          NumPoly,MaxStrip,Bottom,Top,
          StripHeight;
;

float     GazeX,GazeY,GazeZ,EyeX,EyeY,EyeZ,
          AngView,
          WE11,WE12,WE13,WE21,WE22,WE23,
          WE32,WE33,WE43,ViewDist;

float     huge
          ZBuffer[STRIPWIDTH][STRIPHEIGHT];

unsigned char Stg[256];

void  CalcPlaneCoefs(int i);
void  CalcViewCoeffs(void);
void  ClearZBuffer(void);
int   CompareXns(const void *i,
                           const void *j);
void  DoPerspective(float x,float y,
            float z,int *scnx,int *scny);
void  Filled3DPolygon
                (struct PolyStruct pol);
void  GetCentreOfGaze(void);
void  GetData(void);
void  GetEyePoint(void);
void  GetGraphics(void);
void  SetUpGraphics(void);
void  TransformWorldToEye
          (float xw,float yw,float zw,
           float *xe,float *ye,float *ze);
//////////////////////////////////////////

void main()
{ unsigned int n,pol,strip;
  float vx,vy,vz;

  GetGraphics();
  do
  {
  // Raw xyz data in Vertex[].X etc,
  // plane eq and colour in Poly[]
    GetData();
    if (NumPoly!=0)
    { GetCentreOfGaze(); GetEyePoint();
      CalcViewCoeffs();
      SetUpGraphics();
      for (n=0;n<NumVertex;n++)
      { TransformWorldToEye(
            Vertex[n].X-GazeX,
            Vertex[n].Y-GazeY,
            Vertex[n].Z-GazeZ,&vx,&vy,&vz);
        // Vertex[] now eye-space coords
        Vertex[n].X=vx; Vertex[n].Y=vy;
        Vertex[n].Z=vz;
      }
      // equation of plane in eye-space
                          coordinates
      for (pol=0;pol<NumPoly;pol++)
        CalcPlaneCoefs(pol);
      Bottom=VRes-1;
      Top=Bottom-STRIPHEIGHT+1;
      for (strip=0;strip<MaxStrip;strip++)
      {  ClearZBuffer();
         for (pol=0;pol<NumPoly;pol++)
           Filled3DPolygon(Poly[pol]);
         Bottom=Top-1;
         Top=max(Top-STRIPHEIGHT,0);
      }
      getch();
    }
    EndGraphics256();
  } while (NumPoly!=0);
}
//////////////////////////////////////////

void    CalcViewCoeffs(void)
 { double rxy, rxyz, costh,
          sinth, cosph, sinph;

   rxy=sqrt(EyeX*EyeX+EyeY*EyeY);
   rxyz=
     sqrt(EyeX*EyeX+EyeY*EyeY+EyeZ*EyeZ);
   if (rxy==0.0)
   { costh=1.0; sinth=0.0;
   }
   else
   { costh=EyeX/rxy; sinth=EyeY/rxy;
   }
   if (rxyz==0.0)
   { cosph=1.0; sinph=0.0;
   }
   else
   { cosph=EyeZ/rxyz; sinph=rxy/rxyz;
   }
// Coefficients of world- to eye-space
// transformation equations
// [EyeX,EyeY,EyeZ]=[x,y,z,1][WE]
// where coefficients of matrix WE are:

   WE11=-sinth; WE12=-cosph*costh;
                    WE13=-sinph*costh;
   WE21=costh;  WE22=-cosph*sinth;
                    WE23=-sinph*sinth;
               WE32=sinph; WE33=cosph;
                          WE43=rxyz;
// Viewing distance
   ViewDist=0.5*
       HRes/tan(0.5*PION180*AngView);
}
//////////////////////////////////////////

void    CalcPlaneCoefs(int p)
{ int   i;
  float xi,yi,zi,sx=0.0,sy=0.0,sz=0.0,
          xx=0.0,yy=0.0,zz=0.0,
          xy=0.0,yz=0.0,zx=0.0;
  float A,B,C,D,S;

// First reduce n data triples to 3
// by pre-multiplying by the matrix's
// transpose
  for (i=Poly[p].StartVertex;
       i<=Poly[p].FinishVertex;i++)
  { xi=Vertex[i].X; yi=Vertex[i].Y;
    zi=Vertex[i].Z;
    sx+=xi; sy+=yi; sz+=zi; xx+=xi*xi;
    yy+=yi*yi; zz+=zi*zi; xy+=xi*yi;
    yz+=yi*zi; zx+=zi*xi;
  }

// Now finds coeficients of the equation
// of plane ax+by+cz+d=0

  A=+xy*yz*sz+yy*zz*sx+yz*zx*sy -
          yz*yz*sx-zz*sy*xy-yy*zx*sz;
  B=-xx*yz*sz-xy*zz*sx-zx*zx*sy +
          zx*yz*sx+zz*sy*xx+xy*zx*sz;
  C=+xx*yy*sz+xy*yz*sx+zx*xy*sy -
          zx*yy*sx-yz*sy*xx-xy*xy*sz;
  D=-xx*yy*zz-xy*yz*zx-zx*xy*yz +
          zx*yy*zx+xy*xy*zz+xx*yz*yz;

  S=A*A+B*B+C*C;
  if (S!=0.0)
  { S=sqrt(S);
    if (D<0.0) S=-S;
    Poly[p].a=A/S; Poly[p].b=B/S;
    Poly[p].c=C/S; Poly[p].d=D/S;
  }
}
//////////////////////////////////////////

void    ClearZBuffer(void)
{ int   x,y;

  for (x=0;x<STRIPWIDTH;x++)
    for (y=0;y<STRIPHEIGHT;y++)
      ZBuffer[x][y]=FARDISTANCE;
}

int     CompareXns(const void *i,
                   const void *j)
{ if (*((int *)i)<*((int *)j)) return(1);
  if (*((int *)i)>*((int *)j)) return(-1);
  return(0);
}
//////////////////////////////////////////

void    DoPerspective(float x,float y,
                      float z,
                      int *scnx,int *scny)
{ *scnx=ViewDist*x/z; *scny=ViewDist*y/z;
}
//////////////////////////////////////////

void    Filled3DPolygon
                    (struct PolyStruct pol)
{ int  i,n,swapped,v,ymin,ymax,
       x0,y0,x1,y1,y,dy,temp,xx,yy;
  float d,z,dz;

// Build edge list
// Force y-values to be even, will scan
// on odd lines to avoid scan lines going
// through vertices

DoPerspective(Vertex[pol.FinishVertex].X,
              Vertex[pol.FinishVertex].Y,
              Vertex[pol.FinishVertex].Z,
              &x1,&y1); y1*=2;
  MinY=MaxY=y1; NumEdges=0;
  for (v=pol.StartVertex;
       v<=pol.FinishVertex;v++)
  { DoPerspective(Vertex[v].X,
                  Vertex[v].Y,
                  Vertex[v].Z,&x0,&y0);
    y0*=2;
    swapped=FALSE;
    if (y1<y0)
// Force y1>y0 for finding intersections
    { temp=x0; x0=x1; x1=temp;
      temp=y0; y0=y1; y1=temp;
      swapped=TRUE;
    }
    if (y0<MinY) MinY=y0;
    if (y1<MinY) MinY=y1;
    if (y0>MaxY) MaxY=y0;
    if (y1>MaxY) MaxY=y1;
    dy=y1-y0;
    if (dy!=0) /* skip if horiz */
    { d=(((float)(x1-x0))/((float)dy));
      PolyEdge[NumEdges].StartY=y0;
      PolyEdge[NumEdges].FinishY=y1;
      PolyEdge[NumEdges].X=x0+d;
      PolyEdge[NumEdges].DeltaX=2.0*d;
      // For the even/odd line trick
      if (NumEdges<MAXEDGE-1)
      { NumEdges++;
      }
      else
      { DrawString256("Too many edges\
                 in polygon",0,0,7,0);
      }
    }
    if (!swapped)
    { x1=x0; y1=y0;
    }
  }
```

Figure 4 - An implementation of the Z-Buffer algorithm

```
    if ((MinY<(VRes-1))&&(-VRes<=MaxY))
    { MinY=max(MinY,-VRes);
      MaxY=min(MaxY,VRes-2);
  // Find intersections
      for (y=MinY+1;y<MaxY;y+=2)
      { NumXns=0;
        for (n=0;n<NumEdges;n++)
        { if ((PolyEdge[n].StartY<y)
                &&(y<PolyEdge[n].FinishY))
          { Intersection[NumXns]=
                      PolyEdge[n].X;
            PolyEdge[n].X+=
                      PolyEdge[n].DeltaX;
            NumXns++;
          }
        }
        if (NumXns>0)
  // Sort and draw
        { qsort(Intersection,NumXns,
                 sizeof(int),CompareXns);
          for (i=0;i<NumXns-1;i+=2)
          { yy=CenY-(y>>1);
            if (Bottom>=yy && yy>=Top)
            { x0=CenX+Intersection[i];
              x1=CenX+Intersection[i+1];
              if (x0>x1)
              { temp=x0; x0=x1; x1=temp;
              }
              z=((pol.a*(x0-CenX)+pol.b*y)
                  /ViewDist+pol.c)/pol.d;
              dz=pol.a/ViewDist/pol.d;
  // z is -(1/depth) so far things are small
  // negative, near ones large negative.
  // ZBuffer is initilized to 0 so things to
  // be drawn are more negative, ie less
  // than ZBuffer.
            for (xx=x0;xx<=x1;xx++)
              {if (ZBuffer[xx][yy-Bottom]>z)
                { ZBuffer[xx][yy-Bottom]=z;
                SetPixel256(xx,yy,
                            pol.Colour);
                }
                z+=dz;
              }
            }
          }
        }
```

```
      }
    }
  }
  //////////////////////////////////////////

  void    GetCentreOfGaze(void)
  { printf("Centre of gaze ");
    scanf("%f %f %f",&GazeX,&GazeY,&GazeZ);
  }
  /////////////////////////////////////////

  void    GetData(void)
  { int   i,j,nv;
    unsigned char *stg;

    NumVertex=0;
    printf("Number of polygons ");
    scanf("%i",&NumPoly);
    if (NumPoly!=0)
    { for (i=0;i<NumPoly;i++)
      { printf("Polygon %i:\n",i);
        Poly[i].StartVertex=NumVertex;
        printf("Colour ");
        scanf("%i",&Poly[i].Colour);
        printf("Number of vertices ");
        scanf("%i",&nv);
        printf("x y z of vertices:\n");
        for (j=0;j<nv;j++)
        { scanf("%f %f %f",
                &Vertex[NumVertex].X,
                &Vertex[NumVertex].Y,
                &Vertex[NumVertex].Z);
          NumVertex++;
        }
        Poly[i].FinishVertex=NumVertex-1;
      }
    }
  }
  //////////////////////////////////////////

  void    GetEyePoint(void)
  { printf("Eye point and angle of view ");
    scanf("%f %f %f %f",
          &EyeX,&EyeY,&EyeZ,&AngView);
  }
  //////////////////////////////////////////
```

```
void    GetGraphics(void)
{ printf("Type of video adaptor:\n");
  printf("0Video-7,1Paradise,\
2ATI Wonder,3Tecmar (Tseng 3000),4SOTA,\
5Orchid Pro designer II (Tseng 4000),\
6EIZO ");
  scanf("%i",&Type);
  printf("Pixels per line  ");
  scanf("%i",&HRes);
  VRes=3*(HRes)/4; CenX=HRes>>1;
  CenY=VRes>>1;
}
//////////////////////////////////////////

void    SetUpGraphics(void)
{ union REGS regs;

  // some cards need this extra kick
  if (Type==5)
  { switch (HRes)
    { case 1024: regs.x.ax=0x38; break;
      case 800:  regs.x.ax=0x30; break;
      // assume HRes640
      default:   regs.x.ax=0x2e;
    }
    int86(0x10,&regs,&regs);
  }
  InitGraphics256(Type,HRes);
  SetDefaultPalette256(1.0);
  switch (HRes)
  { case 1024: MaxStrip=11; break;
    case 800:  MaxStrip=9; break;
    default:   MaxStrip=7; break;
  }
}
//////////////////////////////////////////

void  TransformWorldToEye
      (float xw,float yw,float zw,
      float *xe,float *ye,float *ze)
{ float x,y,z;

  x=xw-EyeX; y=yw-EyeY; z=zw-EyeZ;
  *xe = WE11*x+WE21*y;
  *ye = WE12*x+WE22*y+WE32*z;
  *ze = WE13*x+WE23*y+WE33*z+WE43;
}
```

Figure 4 - An implementation of the Z-Buffer algorithm

70000 floats for the buffer. This narrow buffer is used to process a strip of the image starting from the bottom of the screen. When all the work in the strip is finished the buffer is cleared and the next strip processed.

A lot of time is spent working out depths. An efficient way to do this is first to use the equation of the plane of the polygon to find the depth at the leading edge where the scan line first cuts the polygon.

The rest of the depths can then be found simply by adding a constant increment at each step. It turns out, for the code given here, that working with -1/*depth* is rather convenient. Near things are then associated with large negative values, far thing with small negative values, the buffer being initialised to 0.0.

The code expects to be given plane polygons, ie ones which have all their vertices lying in the same plane, though their orientation in space can be arbitrary. If you feed it with twisted polygons the effect is likely to be 'interesting'! You can choose any centre-of-gaze and eye-point (as long as they are not exactly on a polygon) and angle of view. Units are approximately millimetres on typical 14" monitors, so a centre-of-gaze at 0,0,0 and eye-point at 300,0,0 with an angle of view of 45° (the whole thing entered as 300 0 0 45) is a good starting point.

There are several improvements which can easily be made to the overall performance of the code

- Make the strip taller if you can spare the bytes.
- Make the strip narrower and taller if you don't require a 1024 pixel width. In each case make appropriate adjustments to STRIPHEIGHT, STRIPWIDTH and MaxStrip so as to reduce the number of slices into which the image is divided.
- Add to the structure PolyStruct the highest and lowest pixel rows which cut a polygon. It is then easy to decide that a polygon is not visible in a particular strip and so does not require further consideration.
- Introduce back-plane culling.

Try it out

Figure 2 provides some test data, comprising overlapping and inter penetrating shapes, to enter after the video card and horizontal resolution has been selected. The program should output a solid shaded image, a drawing of which is shown in Figure 3. Program prompts are in upright characters, suggested responses in italics. The program itself is in Figure 4.

EXE

Dr Graeme Webster was formerly Head of Department of Computer Science and later Deputy Director, Academic, of Teesside Polytechnic. He has been involved with computer graphics for the last 20 years with an especial interest in 3D visualisation for Designers. Currently he is Senior Research Fellow at Teesside Polytechnic working on image processing linked to expert systems and neural networks.

Lazy typists may obtain this code, plus that from Dr Webster's previous articles, by sending in a disk as per the rules laid out on Page 1, column 1. Please mark your envelopes '3D-GRAPHICS'.

Why Choose?

Between Relational and Network when Raima Data Manager has both and now MUCH MORE

The Raima Data Manager dbms is intended for use by the professional C applications developer. Raima is written in C and provides a complete set of sophisticated development tools that feature:

- High speed access to large or complex data
- Mainframe functionality
- Portability to any C environment
- Royalty-free run-time distribution
- Source code available in C.

Raima provides both Relational and Network model technology for programming flexibility. Retrieve a record using the relational keyed access method and all related records can be immediately available using the network database model.

db_QUERY provides SQL-like access to **Raima**, callable from within an application, **db_REVISE** allows restructuring of live databases.

Raima is fast. Access times for retrieving data are largely independent of the size of the database.

Raima is portable. Develop an application in any environment and your database access code can be ported without change to any other supported environment. Currently, **Raima** supports MSDOS, MS Windows, OS/2, QNX, UNIX System V and Berkeley, Ultrix and VMS.

Raima Data Manager dbms is a product of Raima Corporation and has been sold to thousands of C programmers in over 50 countries. **Raima** is fully supported in the UK and Eire by Systemstar Ltd. Comprehensive training is available together with bespoke system development. At Systemstar we offer **C-scape 3**® screen interface library with Look and Feel™ from Liant Software Corporation to provide **Raima** programmers with a complete application development environment.

For more information about Raima Data Manager dbms, db_QUERY, db_REVISE and C-scape 3 call Systemstar in Hertford on (0992) 500919.

FEATURES	YES	NO
Single and multi-user available	✓	
Relational B-tree indexing	✓	
Network database model	✓	
Multiple database access	✓	
Referential integrity	✓	
Transaction processing	✓	
Automatic recovery	✓	
Record and file locking	✓	
RAM resident		✓
Relational Query and report writer	✓	
Total database redesign/restructuring	✓	
C compilers: most supported	✓	
C ++ compatible	✓	
Operating Systems: VMS, Ultrix,	✓	
UNIX, BSD, SunOs, QNX	✓	
MSDOS, OS/2	✓	
Windows 3	✓	
LANs: Netbios and NFS	✓	
Read and write WKS, WK1 and DBF files*	✓	
Source Code available	✓	
Training courses available	✓	
Run-time Royalties (Absolutely NOT)		✓
*using WKS Library		

NEW SHOWCASE

Support for NFS
Object Manager Class Library
Protected Mode for **MS-DOS**
VISUAL BASIC support for Windows

SYSTEMSTAR
LIMITED

1-3 PARLIAMENT SQUARE HERTFORD SG14 1EX
TELEPHONE: (0992) 500919 FACSIMILE: (0992) 554261

CIRCLE NO. 844

Are Hackers Really Criminals?

Last August the Computer Misuse Act came into being. David Martin reviews the current attitude of the law to Computer Security, and asks what we should be doing about it.

For many years there has been no legal recourse when a computer owner has found that someone has been tampering with the information in his computer system. This ended last August, with the introduction of the Computer Misuse Act 1990. However the law is open to wide interpretation and, as a result of recent cases, there are specific areas that need to be carefully considered. But the law isn't one sided, there are requirements within the Data Protection Act placed upon the manager of a computer system to ensure that personal information is not disclosed for unlicensed purposes. Furthermore, the Police and Criminal Evidence Act is very strict about the use of computer records as evidence in court. This is also the responsibility of the computer manager and the system designer.

History

In the United Kingdom, the theft of information is not a crime. You can go into a shop, pick up a newspaper, read it and then place it back on the shelf. You are free to leave the shop and owe nothing. So it has been with software. A hacker (for the purposes of this article I shall define a hacker as someone who is accessing a computer system in a way that would certainly cause concern to the owner of the system) who has broken into a computer system has been free to read information and copy it back to his own system, either via a modem link or by use of 'SneakerNET'.

The police in many countries havé attempted to respond to these activities by the use of inappropriate laws such as the Theft Act, where the perpetrator has been charged with stealing the electrons that have flowed down the telephone line. Where data has been modified, then the offence of Criminal Damage has been cited. However, the courts have not appeared to have lived up to their responsibility to deter the arrogant vermin in our society.

Something needed to be done to correct this situation. Thanks to MPs such as Emma Nicholson, we now have a Computer Misuse Act in place. No-one would say that it is perfect. However, it is a very good first step. It will be amended in time by the rulings of the courts and by specific amendments that Parliament feels necessary. The computer system manager can sleep safely at night knowing that all unauthorised users would not dare to access his system as the full weight of English Law will protect him. Right? *Wrong...*

Digital publicly admitted that its security had been breached. It also proved that it would not tolerate such an attack

There are three main sections within the Computer Misuse Act that deals with the new crime definitions and many sections in both the Data Protection Act and the Police and Criminal evidence Act that do not afford the protection we deserve.

The Crimes

The Computer Misuse Act has three distinct sections defining the offences. In broad terms they criminalise unauthorised access, use of a computer system to commit an offence and causing the computer system to perform unauthorised actions.

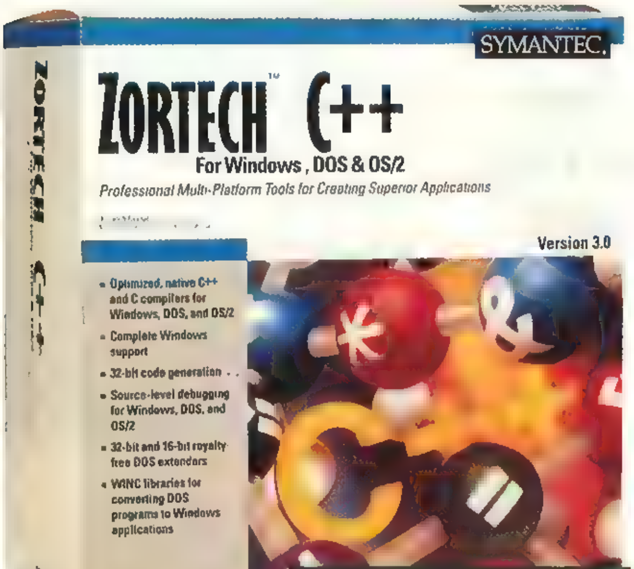
The first offence of unauthorised access carries a maximum prison sentence of up to six months, and a 'scale five' fine, this is currently up to £2000. The second offence is the use of a computer system to commit a further offence. This could be a fraud involving the transfer of funds, or even the use of a computer system to commit murder. Just think of it: a hacker could break into a hospital system and change drug doses for a specific patient. The changed doses may mean that a particular person does not get enough life-saving medication, or he receives an overdose. All done without the hacker having to move from the comfort of his PC and modem at home. In this particular case, the culprit is likely to be charged with murder and the fact that the blunt instrument was replaced by a microprocessor would probably be ignored.

The third offence, like the second, carries a maximum of a five year prison sentence and an unlimited fine. This is the offence of causing a computer system to perform unauthorised actions. This includes the modification or copying of data, execution of programs and even requesting a printout. Once again our potential murderer would have committed this offence.

The big problem is *What is unauthorised access?* There are two schools of thought. The first states that everything is permitted unless explicitly prohibited. The second is that everything is prohibited unless explicitly permitted. However the law takes a more reasonable view. There was a case brought to an industrial tribunal in 1978 where two employees had been fighting. There was nothing in the company rulebook to say that fighting was prohibited. However the Mr Justice Kilner Brown decided that certain actions could reasonably be thought 'to be regarded very gravely by management' that 'it ought not to be necessary for anybody ... to have [it] in black and white in the form of a rule'.

THE PROFESSIONAL PROGRAMMERS' CHOICE

Zortech C++ for Windows, DOS, OS/2 & UNIX.


SYMANTEC.
ZORTECH™ C++
For Windows , DOS & OS/2
Professional Multi-Platform Tools for Creating Superior Applications
Version 3.0
• Optimized, native C++ and C compilers for Windows, DOS, and OS/2
• Complete Windows support
• 32-bit code generation
• Source-level debugging for Windows, DOS, and OS/2
• 32-bit and 16-bit royalty free DOS extenders
• WINC libraries for converting DOS programs to Windows applications
ZORTECH C++

Now the world's first multiple platform C++ compiler is available from Symantec – Zortech C++ version 3.0 Release 4.

When it comes to building large, complex applications, C++ is the programmers' language of choice.

Zortech C++ for Windows, DOS & OS/2 is the only professional compiler to offer tools for Windows, DOS and OS/2 in an integrated package. Zortech C++ goes well beyond other compilers presenting not only professional tools but a collection of exclusive features specifically designed to help you increase your productivity and reduce your development time.

Zortech C++ offers a WINC library that enables you to easily convert DOS command line programs to true Windows applications. And you won't need to switch development systems, because you'll have the flexibility of moving across different platforms.

All of which means you'll get your applications to market faster than ever before.

"All in all Zortech has put together a winner at every level and offers a lot of compiler for the money". **– Don Binns, Program Now, June 1991**

Developer's Edition **£399.00*** Science & Engineering Edition **£599.00***
UNIX Edition **£399.00*** Macintosh Edition **£249.00***

For immediate information use your fax telephone to call Facts Direct on 0865 727232.

When asked for a six digit code enter (code number) or 202024 for your price list.

≡FACTS DIRECT

PHONE **0865 727 232**
FROM YOUR FAX MACHINE
HANDSET – WHEN ASKED
FOR A CODE ENTER:

◆

202 021 FOR **APPLICATIONS**

◆

202 022 FOR **UTILITIES**

◆

202 023 FOR **MAC UTILITIES** AND **GREATWORKS**

◆

202 024 FOR **THE SYMANTEC PRICE LIST**

PROVEN GLOBAL OPTIMISER – Zortech C++ offers globally optimised compilers that boost program speed and significantly reduce program size. So regardless of the platform, Zortech C++ delivers performance and stability where it matters – in your finished user application.

DOS EXTENDERS – Zortech C++ even includes royalty free 32 bit and 16 bit DOS extenders, for the cost effective development of programmes with advanced memory requirements.

WINDOWS – Zortech C++ provides absolutely everything you need to edit, compile, link and debug any Windows application you won't need to buy Windows SDK. We've also added a help compiler and engine, a resource compiler, Windows header files and libraries, and printed SDK documentation.

TOOLS AND UTILITIES – with source code provided – help avoid re-inventing the wheel e.g. TSR's mouse drivers, memory tools. A list of 3rd party products which support Zortech C++ is included with the product, for example ZINC interface library and POET object-orientated database library.

FULL SOURCE CODE – to all standard libraries included.

Call NOW for more information

0628 777677

SYMANTEC.

Fax No: 0628 776775

I want the professionals choice for a professional result!

□ **Please send me more information on Zortech C++ NOW!**

Name Position

Company

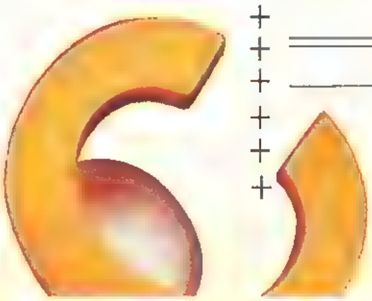
Address

Tel No.

.EXE 4/92

Symantec (UK) Ltd, MKA House, 36 King Street,
Maidenhead, Berkshire, SL6 1EF

CIRCLE NO. 845



This was brought up during an Employment Appeals Tribunal hearing recently where a shop steward had been using a computer system in the middle of the night. The shop steward had already got an account of his own. However, by use of a password used by his daughter, he accessed information that he was not required, by his job, to be able to access. The Tribunal decided that although the employer should have defined exactly the extent of access permitted, any reasonable person would have realised that this was indeed unauthorised access.

A computer system manager should therefore ensure that any *Welcome* banner states that if the user does not have explicit permission to access the computer system and use it for an explicitly permitted set of actions, he should log out.

So now we have case history that states what is and what is not unauthorised access. Unauthorised access can be defined as *something which is going to be regarded very gravely by management.* The tribunal also placed a requirement on the owner of the system that rules defining its use ought to be put in writing so that there could not be any confusion in the future.

An interesting thought is that most (over three quarters if various surveys are to be believed) unauthorised access is caused by people who already have a legitimate need to use a particular computer system.

This is particularly true in an academic environment where students are encouraged to discover knowledge by experience. This causes a great deal of problems to the running of an academic system. Therefore users are warned that 'hacking' or other such activities could lead to their computer usage being withdrawn. I know of one particular University that expels a student every three years or so to ensure that they do not suffer too badly from unauthorised activities and that the system remains available to the vast majority of students who need the resources so that they can obtain their degrees. Now that the new Computer Misuse Act is available, they have a second option open to them, whether they decide that this has more effect as a deterrent remains to be seen.

It has been suggested, by some ill-informed sources, that it is the responsibility of a computer system manager and the system designers to ensure that no-one ever breaks into the system. The main problem with high-security systems is that they prevent the legitimate user from performing his work in the most efficient manner. If you want a system to be totally secure then you must switch it off and place it in a locked room! Is this really the way that computers should be operated? Most of us would be far more efficient at work, or in our studies, if all systems were as open as our home PCs. However, when we are working at home the only person who would suffer from an unexpected system crash is ourselves.

One particular University expels a student every three years to ensure that they do not suffer from hacking too badly

Recent Court Cases

There have been a number of cases that have come to court recently. In July 1991, Sean Cropp was found not guilty at Snaresbrook Crown Court by Judge Francis Aglionby. For some unknown reason, the Judge stated that the Section One offence of Unauthorised Access only applied if one computer system was used to gain access to another computer system. Cropp had used a dumb terminal! The only thing that can reasonably be deduced from this case is that the Judge knew either nothing or little about the use of computer systems. It makes no difference to the victim computer if the input is coming from a dumb terminal or from a PC, both are equally damaging. This brings out the point that there is little point in having good computer related legislation if members of the legal profession appear to be unable to understand the reasoning behind the law and the state of technology in use today. In this particular case it appears that something has gone badly wrong.

On a more positive note, in April 1991, Ross Pearlstone was fined £900 for two breaches of the section 2 offence after he accessed Mercury Telecommunications Computer Systems. It appears that the only reason that this case succeeded was that Pearlstone had a fit of conscience and sent a cheque to Mercury for £1600.99 (the amount he had defrauded Mercury by when he made free transatlantic telephone calls).

In the Pearlstone case it was easy to trace who had committed the crime, and for the prosecution to obtain documentary evidence. After all Pearlstone metaphorically put up his hand and said *I did it, please come and arrest me.*

But life isn't always going to be that easy. The Police and Criminal Evidence Act has a section (Section 69) dealing exclusively with evidence from computer records. This allows a court to accept computer records subject to certain conditions. The main conditions are that there is no reason to disbelieve the magnetic record and that the computer system was functioning correctly at the time of the incident. But if the hacker is a Superuser, in UNIX terminology, surely there can be nothing that can be absolutely relied upon in terms of the information on the disk. The hacker has had the ability to change anything he wants to including any audit trails.

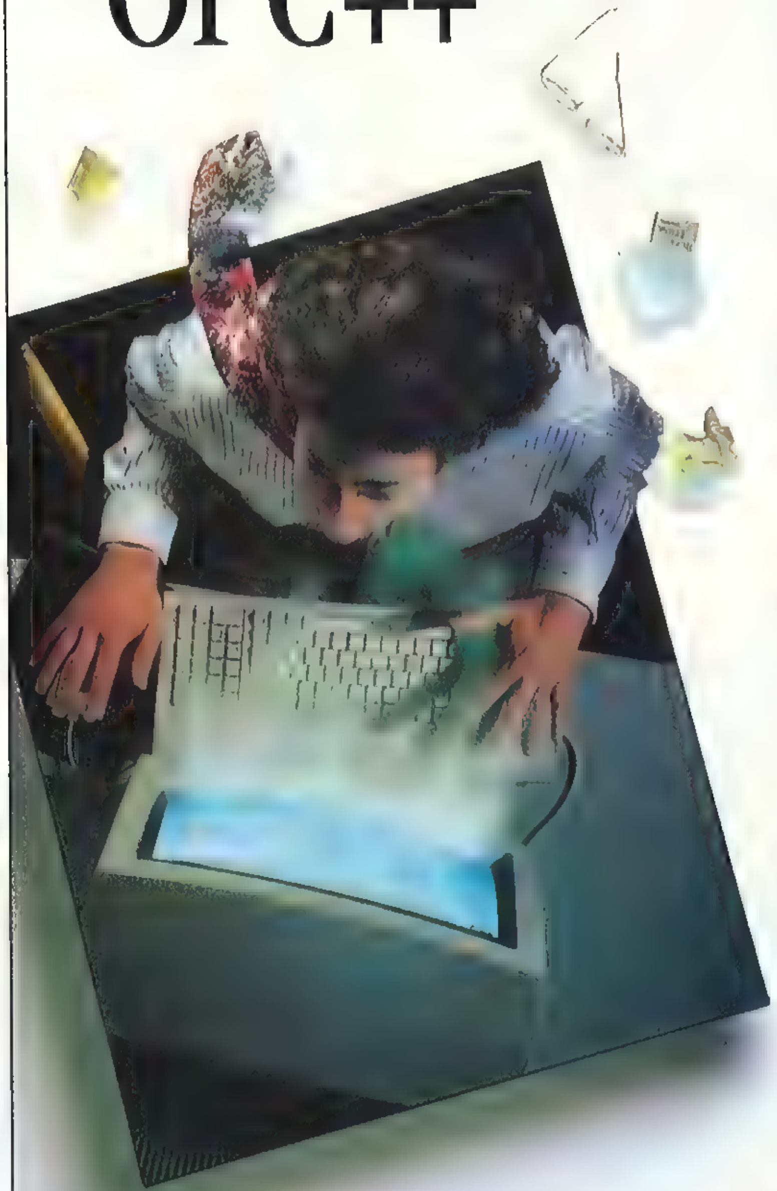
The way in which Section 69 has been interpreted in court, particularly in fraud cases where banking records have been provided from a computer system, is that the system manager, or one of his staff, has been called as a witness to state that he believes that the records produced by the computer, as hard copy, are indeed a true statement of what actually happened.

However, the banks have been unwilling to go to court where a customer has claimed that a 'phantom withdrawal' has been made on an account. Recently I had a withdrawal made against one of my credit cards. Only a small amount, actually it was under £10. I could not have possibly been in the place where the withdrawal was made, as even according to the credit card company's own records I was over a hundred miles away only thirty minutes before the withdrawal. After numerous telephone conversations the amount was credited to my account. I believe that the credit car company would have been very unwilling to contest this situation in court, as a ruling against them would undoubtedly have caused problems as later case law. In fact I know of no case where a bank has actually gone to court rather than make an out-of-court settlement as a 'gesture of goodwill' to the customer.

The US approach

In the States, security has had a far higher profile as the market has been driven by the

Unleash The True Power Of C++



It Takes More Than The Language.

Unleashing the full potential of C + + takes powerful object-oriented tools and a rich library of classes. Using only a C + + compiler and traditional tools yields traditional results – slipped schedules and maintenance nightmares. Only Objectworks\C + +, Release 2.4, provides the comprehensive development environment that allows you to produce extraordinary results.

Speed Your Development and Maintenance.

Objectworks\C + + provides dynamic, graphical browsers to illuminate the tangle of class relationships, object interactions, and program structure, as no static or textual information possibly can. The unique integration of the C + + source level debugger and interactive browsers ensure that essential information is instantly at your finger tips. And, a broad range of cross references are as close as a point and click. All of these tools aid in the understanding of code, increase reuse, decrease confusion, promote consistency and, ultimately, speed your development and maintenance.

An Open Environment.

Based in the latest AT&T C + + language System, Release 2.1, Objectworks\C + + provides support for SunView and X Windows on Sun platforms. Its open environment also allows you to use your favourite C preprocessor, C compiler, linker, profiler, or source code control system. The sophisticated class libraries included in Objectkit\C + + exploit the natural synergy between the development environment and the extensive reusable class libraries. Objectworks\C + + also allows programmers to use existing makefiles without any modification, saving time and money.

Team Programming.

Objectworks\C + + is the only C + + development environment that allows engineers to work cooperatively, loading and browsing one another's code, without interrupting each other. Programmers simply access an information file containing a description of the code they need, and then they query this description.

A I International are uniquely qualified to help you exploit the benefits of object-oriented technology and C + + programming applicable to your development efforts. If you'd like our technical management brief "Exploiting the Full Benefits of C + + Objects", or are ready to unleash the true power of C + +, call us today at 0442 876448

Without Objectworks, you're without objects.

A I INTERNATIONAL LIMITED

The Chapel, Park View House, 1 Park View Road, Berkhamsted, Herts, HP4 3EY
Telephone: 0442 876448 Fax: 0442 877997

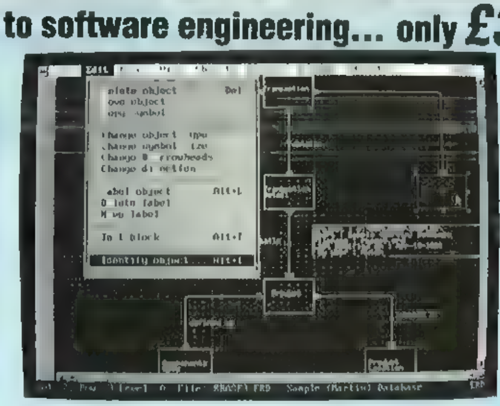
CIRCLE NO. 846

EasyCASE™ Plus

NEW! VERSION 3.0 Call for upgrade details.

The affordable approach to software engineering... only £320*

Finally, there's a CASE tool that won't get in the way of your creativity... A tool that makes structured analysis, structured design and data modeling as easy as working with any other tool on your PC - EasyCASE Plus! Using EasyCASE Plus' new, easy to use graphical user interface (GUI), you'll be creating and editing charts, linking them, and building your data dictionary in no time. As well as being easy to use and easy to learn, EasyCASE Plus is easy on your budget! Ask any user. They'll tell you it's the best buy for your PC based CASE tool needs. Discover why over 4,000 software professionals use EasyCASE Plus and how you can join them!



"EasyCASE Plus is a well designed, low priced tool that is easy to learn and provides excellent diagramming capabilities... EasyCASE Plus is an excellent investment."

COMPUTER LANGUAGE PRODUCTIVITY AWARD 1990

Features:

- IBM SAA/CUA compliant graphical user interface (GUI)
- Extensive diagram editing features
- Integrated dBASE III compatible data dictionary
- Integrated dictionary manager, reports manager, process editor
- Hierarchical chart linking & process decomposition
- Record and element definitions
- Extensive printer, plotter and desktop publishing support
- Data dictionary import, export, and merge
- On-line help
- Comprehensive documentation with tutorial
- Access to your database, word processor, DOS, etc.
- Integrated diagram analysis (optional)

Requirements:

Runs on: IBM PC or PS/2 (AT recommended), DOS 3.1 or higher, EGA/VGA color, mouse, 640 K RAM (500 K free), 1 MB EMS recommended, math co-processor supported. Printers/Plotters Supported: Epson FX & LQ, IBM Graphics & Proprinter X24, HP QuietJet, DeskJet, & LaserJet, HP Plotters, PostScript.

EasyCASE Professional..... £420*
(includes integrated DFD level balancing and data dictionary/diagram analysis)

Methods:

- Yourdon/DeMarco
- Gane & Sarson
- SSADM (DFDs)
- Ward-Mellor/Hatley
- Yourdon/Constantine
- Martin
- Chen, Bachman

Diagram Types:

- Data Flow Diagrams (DFDs)
- Structure Charts
- State Transition Diagrams
- Entity Relationship (ERDs)
- Data Model Diagrams
- Transformation Schema (real-time DFDs)

Evergreen CASE Tools

THE SOFTWARE CONSTRUCTION CO. LTD.
FAX: (0763) 244025

MasterCard VISA

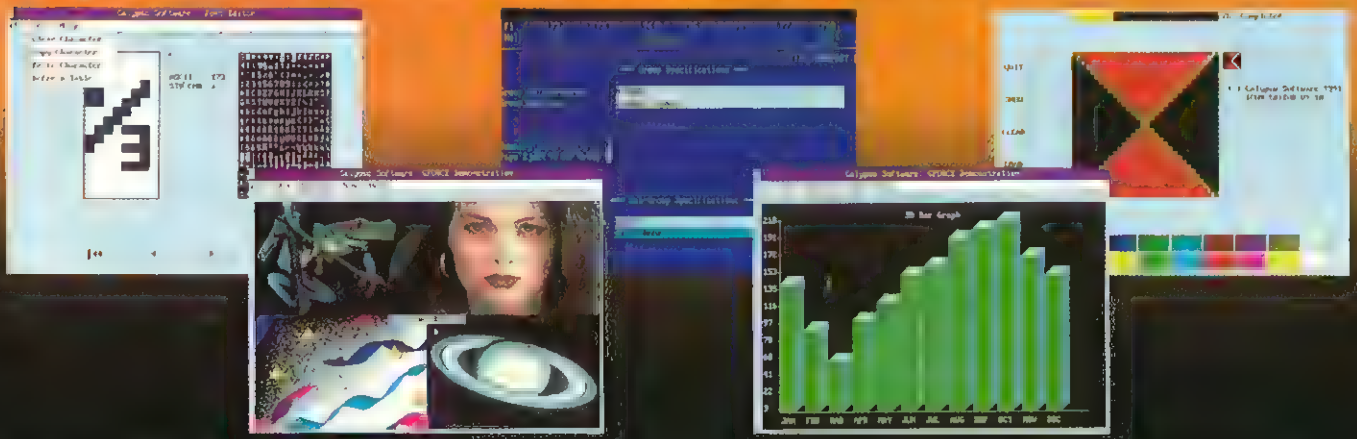
Call today for a brochure!
Tel: (0763) 244114

©1991 by Evergreen CASE Tools, Inc., All Rights Reserved.

1 The Maltings, Green Drift, Royston, Hertfordshire SG8 5DB

CIRCLE NO. 847

Spot the difference...



Calypso Software GFORCE Demonstration
3D Bar Graph
JAN FEB MAR APR MAY JUN JUL AUG SEP OCT NOV DEC

All of these screens have been taken from Clipper applications. The difference is that four of the screens use the GFORCE graphical user interface. GFORCE provides a very fast Windows like environment as a linkable library for the Clipper compiler. GFORCE includes low level functions for pixels, lines, boxes, circles, fills, bitblits and high level functions for 3D buttons, pull down menus, dialogs, scroll bars, icons, bitmaps and fonts. GFORCE retails for £185.00 including full documentation, Norton Guides, Quick library with complete Clipper source graphical font and icon editors with Clipper source and ninety days technical support.

GFORCE works with Clipper® 5.01, Summer '87 and overlay linkers such as Blinker® etc.

Please call for a free demo disk.

QBS Software Ltd
10 Barley Mow Passage
London W4 4PH, ENGLAND
Tel: 081 994 4842
Fax: 081 994 3441

CIRCLE NO. 848

GFORCE™ Clipper Graphical User Interface

Department of Defense and the NCSC (National Computer Security Center). Together they have produced the 'Trusted Computer System Evaluation Criteria'. It is normally referred to as 'The Orange Book' because it has a bright orange cover. This book defines various criteria against which the security of a particular system, both hardware and software can be checked. The security rating range from 'D', which means minimal protection such as a PC - where all you need to do to access everything is to find the on/off button, through to 'A1', which is verified design. There is a Honeywell system that actually meets the 'A1' requirement, however it cannot be described as either the cheapest or easiest system to use.

This is the crux of the matter. It is possible to engineer systems that are very secure, but they cost so much that they are available to only a chosen few, and they can be very difficult to use.

When industry and academic computer departments become worried about the unauthorised access to their systems, they have very few choices. If security is going to be improved, then either the system will not be available to so many people or the system will not be as easy to use. Hackers have claimed, in the past, that their actions are constructive in that they point out the security weaknesses of computer systems. This is utter rubbish. I do not need someone with a sledge-hammer at my front door to prove that it could be improved by the fitting of a steel door with a moat and drawbridge to protect it. Likewise, computer designers do not need the antics of system vandals to point out that security could be improved.

All these vandals do is gradually make it more difficult for the rest of us to do the work that we should be doing. If it takes longer to do because of the security required then the company is less profitable. If the company is less profitable then it cannot afford to pay us as much. Ultimately these reprobates are being funded by everyone who works in the computer industry through our pay packets. But we live in a real world, the undesirable elements of our society do exist. As a computer service provider we need to improve our system security to a point whereby the expected loss from an attack becomes manageable.

What should we do

So what can we as a community do? Initially by our example of working in a professional manner and treating the miscreants with the disdain that they deserve. If we discover that someone is either hacking or doing anything similar then it should be reported to an appropriate authority. The Digital Equipment Corporation took a very notable stand when it decided to prosecute Robert Mitnik in the States. One of the world's largest computer manufacturers publicly admitted that its security had been breached. It also proved that it would not tolerate such an attack on any of its systems. This should serve as an example to the industry as a whole.

> ***Computer designers do not need the antics of system vandals to point out that security could be improved***

So why don't all companies resort to the use of the law when it has been specifically designed to protect their computer systems? In the past there have been stories of hackers being offered, by the company, immunity from prosecution if they identify the security loophole and therefore prevent others from entering via that route. Also some companies, particularly financial institutes, believe that the possible bad publicity of the public statement, in court, that their computer system had been 'cracked' would cause serious business problems.

One question that should be addressed concerns situations where computer equipment has been used in a crime, and can be seized by the police as evidence for court proceedings. If the computer system belonged to a company and an employee was misusing the system without the company's knowledge, 'Is it fair that a company could legally be deprived of its computing resources by the actions an employee committing a crime?' In practice, the police are likely to collect all the evidence that they require, even when this causes the system to be unavailable, and then return its use back to the company, so long as the company is not suspected of committing a crime, and they have been co-operative during the investigation phase.

In the case of a home system that has been used by the owner to commit a crime, is it reasonable that the system should be returned after conviction, or should the courts normally make an order for the system to be forfeit? This has yet to be decided.

Ethics Module

It has been suggested, by the National Computer Users Forum (NCUF), that an ethics module should be included in all academic and industrial computer training. This should highlight the consequences of the inappropriate use of a computer system. In the same way as joy riders need to be made aware of the consequences of their actions, so should it be with computers. Hopefully this good intention will produce effects in the long term.

If you are the manager of a computer system, reappraise your requirements for computer security. Make it appropriate to the possible loss not only of your data, but the downtime that may ensue. For this to be effective it is essential that you have the backing of the most senior level of management you can lay your hands on, because good security always costs money.

If you are a user, make the system manager aware of any loopholes that you discover. Remember that a bug in a program may be able to wipe data even faster than any hacker can.

So, in summary, there are now specific criminal offences that can be used to convict the errant members of our society. But these laws are still in their infancy. There is no substitute for good computer security as finger-trouble will always occur so long as there is a human anywhere in the loop.

The future of open, easy to use systems is now protected by law, but it is the responsibility of us all to stop it having to be used after the damage has been done.

David Martin is a computer consultant who has been working with secure computer systems since the late 1970s. He is employed as a Consultant with C.O.R.A.L. and has provided advice and expertise to numerous clients in the commercial, defence and industrial sectors. David is also an active member of the DECUS (Digital Equipment Corporation Users Society) Security Group and has presented Computer Security related papers to many technical and management forums.

A marriage made in heaven?

Since Microsoft broke up with IBM, it has been looking for a new trading partner. Jules has been speaking with someone who may know who the partner is to be.

The Americans have always had a soft spot for religion - their churches invented Gospel music, their fundamental Christian preachers are unique the world over, and even Nancy Reagan consulted an astrologer on her husband's behalf before allowing him to take any decisions (the revelations of which were greeted with far more incredulity over here than over there, by the way).

With religion so much a part of American life, it should come as no surprise that Microsoft, who will gladly take advantage of almost anything to help achieve their ends, was religiously motivated in their appointment of a new man. He has already had a significant influence on the company - the tagline 'Making it all make sense' which has appeared on their products for years is being reviewed. The original plan was that, since nobody believed the line anyway, it would be changed to 'Making it all make money', but this new man suggested merely 'Making it all' as being more accurate and describing a better direction for the company.

The new man is the Vice President in charge of World Domination, and his name is Bill Z. Bubb. Although Bill himself says that he has been around a long time, working 'behind the scenes', it is not clear on which projects, or for which companies. He says, obliquely, that he was instrumental in placing computers in the utilities companies in the fifties and sixties (the ones that sent out final demands for £0.00). What he does claim is that he has some very special contacts, and a unique assortment of skills which he can bring to bear. He also has what he cryptically describes as an international group of companies, which has no intrinsic trade, but facilitates other companies in their endeavours. Among its assets is a medical company which developed a drug treatment to confer eternal youth, a number of banking interests, and a small furry animal disposal business.

Here is a man who controls an organisation spanning the entire globe - and beyond

How does one become a Veep at Microsoft without any significant public reputation? Mr Bubb explains 'Bill and I have known each other for years, and I have often given him advice and other services - he was very interested in my medical company, for example. Microsoft is committed to the C language (in which I played a part), and the company has reached a stage in its development where it seemed appropriate that I should join the board.' In fact, he dismisses the suggestion that he operates invisibly. 'Within my own field, I do have something of a reputation. I have always dealt fairly with my clients, and invariably explain my terms of business before any goods change hands. I have never reneged on a promise. My clients respect that, so my businesses have succeeded, in spite of what some people perceive to be high prices.'

What of the future? Mr Bubb was hired to boost Microsoft's sales, and he has some very clear ideas about how that should be done. 'I have always believed in giving a decent service. Give the customer what he wants, and keep him happy while he uses it, and he will never approach the competition. As a company, Microsoft makes great products, and then doesn't follow through with the service and support. I've tried that and it doesn't work - you get some sales, but no repeat business. The future is about service.' Off the record, he has a few surprises up his sleeve. It seems he has commissioned some research which suggests that the programming and productivity software published by competing companies is not going to survive. 'The world is changing, as everyone knows. There will come a time, and it is not far off, when the world will change so much that all those compilers released by other companies will simply cease to work. Overnight, the programs created on them will not run on any machines available. Forewarned is forearmed, and we are ready for the changes. Are they?' It's unlikely they are; Mr Bubb was somewhat cagey about the nature of those changes and wouldn't be drawn on who the most exposed companies are. 'We are in a position to protect certain trusted organisations from the worst of it' he said.

He is very clear about the role of software in the computer industry. 'Software is about communication. When you read a book, the author of the book is talking directly to you. We perceive software in the same way, except now there is a dialogue going on - the author is not just speaking to you but is working for you. We want to put a computer on every desk in the world, and we want a bit of Microsoft in every one of them. That way, both Microsoft and my own organisation can talk to, and can work for every human being on the planet. You see, we only want to make people happy!' Fine sentiments indeed - we will see just how far Bill's influence can spread.

The question demands to be asked, is the appointment of Mr Bubb fair trading? After all, here is a man who controls an organisation spanning the entire globe ('and beyond', Mr Bubb adds parenthetically) who is going to put his (not inconsiderable) weight behind one company. 'I don't see why not. I have offered my services to many other organisations, both large and small, and those offers are still open. Some companies are, of course, not in a position to accept, putting their faith in my competition. I think that is a mistake, but I'm sure they are happy in their decision. Hard work and blind faith can work miracles!'

Microsoft did need a new trading partner, and it seems they have found someone who can easily replace the power of IBM. They have some very interesting strategies, and if the plans which Mr Bubb was prepared to talk about are any guide, the plans which are still under wraps are going to be earth-shattering. One can only wait and see what the future will bring.

EXE

Since conducting this interview, Jules has achieved commercial success, .EXE has accidentally increased his pay by four times, and he has become irresistible to women. The sales office of Bill Z. Bubb (UK) Ltd can be contacted on 0707 44185 or on CIX as jules.

War Against Imbecile User Interfaces

Microsoft spent millions of dollars to develop flashy pulldown menus and contoured pushbuttons. But it still can't figure out the difference between a baud rate and a blind date...

You want to see it for yourself? Ok, walk over to your computer, call up Windows, go into Control Panel. Now select 'Ports', 'Settings'. You see there where it asks for baud rate? Want to see something stupid? Type your name. That's right, your name. 'But', you cry, 'that's a numeric field! How can I type alpha there?'. Don't worry, just do it!

'Ayfer... Hey, it's accepting it!' Pretty disgusting, huh? Hundreds of man years of development time, megabytes upon megabytes of source code, tons of books and documentation, and it lets you tell it that you want 'Harry Johnson' as the baud rate for COM2.

It isn't just in Control Panel, either. Aside from making sure the string doesn't grow too long, a large number of Windows applications out there have no input verification at the keystroke level. Not surprising when you consider that the string editor used in Windows dialog boxes (windows of the 'Edit' class) can only check for length and convert to upper or lower case. Any other control must be done specifically by the application.

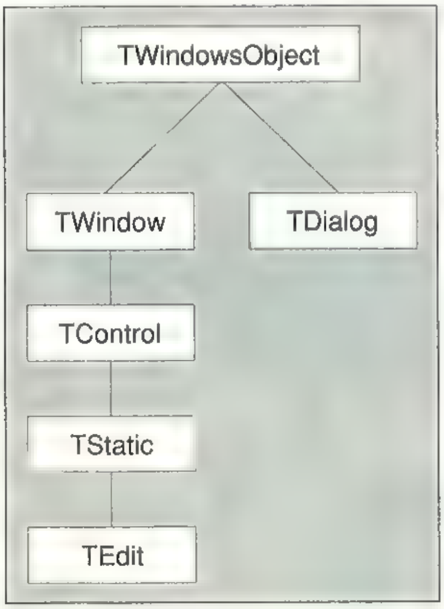


Figure 1 - Part class hierarchy of OWL

I'm not sure why Windows programs end up with such lousy input checking. I suppose the programmers get so caught up in the flash of dialog boxes with bitmapped pushbuttons and popup help menus that they forget about those basic principles of user interface design that existed even way back in dBASE II on the Z80. Or maybe they think that 'User Friendly' means 'Let them type whatever they want'.

In all my earlier attempts at writing Windows programs, I managed to avoid this problem. I played with graphics instead. And a string here and there when necessary. Who needs numbers, when I can show the size by dragging the mouse? But now the time has come. I need to input some numbers. With decimal points even. Just to spice things up, I have spent the previous evening erasing *everything* from my D drive so that I could install most (but not all) of Borland C++ 3.0 and the OWL (Object Windows) libraries. What better way to acquaint myself with OWL than by designing a keystroke verifying Edit class?

The Playing Field

First Stop - the OWL manual introduction chapter to learn about the 'OWL philosophy', then a quick trip through the example programs to find one that uses the `TEdit` class included in OWL. Compile that program, then pop into the class browser to get a graphical look at OWL's class hierarchy.

All OWL Windows are based on the `TWindowsObject` class, and most (except `TDialogs`) are derived through `TWindowObject's` immediate descendant, `TWindow`. The basic functionality needed by any window is defined in these two base classes. See Figure 1 for the derivation path leading to our pet class, `TEdit`.

OWL uses a neat concept called 'DDVT' (Dynamic Dispatch Virtual Tables) to allow you to put the message handler for each message sent to a window into a separate function, rather than forcing you to glop them all into one big `WndProc()` like standard C Windows programs. If you want a certain message to be handled differently, just derive a new class which redefines the DD function in question.

Attempt #1

In our case, what we want is an Edit window which checks input characters. The Windows Reference tells us that characters are sent to a window's `WndProc()` via the `WM_CHAR` message, but since we're using OWL, we just derive a class from the existing `TEdit` which defines a `WMChar()` member function having the DDVT number `WM_FIRST+WM_CHAR` (note it is the DDVT number that is important, not the function name). The `TNumEdit` class in the program in Figure 2 does exactly that - the `WMChar` function checks characters sent to the Edit window and only calls the `DefWndProc()` for 0..9, the minus sign, and the decimal point.

I admit I was surprised when this worked. I would expect these Window procedure interceptions to work on windows of classes defined by the application (I'm speaking of MS-Windows registration classes here - the kind made with the Windows API RegisterClass() function), but TEdit creates windows of the pre-defined class Edit. I had assumed that the original WndProc() for an Edit window would be called, bypassing the DDVT's message trapping. Happily, the OWL designers thought of that; although they use the pre-defined Edit class for TEdit windows, they use a technique called subclassing to redirect WndProc() calls to a special function inside OWL. This function first checks the DDVT table, and calls the original WndProc() if there are no matching DDVT functions. I was so intrigued by this that I recompiled the entire OWL library with debugging turned on so I could trace through and watch what happened.

TNumEdit works - it only allows numbers, minus, and decimal. But it has problems. Most importantly, you can enter as many minuses or decimal points in as many places as you like. Also, I prefer numbers to always appear right justified. Besides, a more general solution would be nice.

Attempt #2 - TMaskedEdit

The TMaskedEdit class (also in the program in Figure 2) is a more general solution. If you declare an object of type TMaskedEdit, it will behave almost identically to a TEdit. It is infinitely more useful if you derive a new class from TMaskedEdit, redefining the Filter() function to decide which characters to accept. (TMNumEdit in the same program is an example of this.)

The internal magic of TMaskedEdit is again in the WMChar() function.

```
// MASKEDIT.CPP - test program for
// TNumEdit and TMaskedEdit classes
// requires Borland C++ v3.0 and OWL
//
//  Righttocopy (r) Laine Stump, 1992
//  - No Rights Reserved
#include <string.h>
#include <owl.h>
#include <static.h>
#include <edit.h>
//****************************************
class TRightStatic : public TStatic
  {  // right justified static text
public:
  TRightStatic(PTWindowsObject AParent,
               int AnId, LPSTR ATitle,
               int X, int Y,
               int W, int H,WORD ATextLen,
               PTModule AModule = NULL)
    : TStatic(AParent, AnId, ATitle, X, Y,
              W, H, ATextLen, AModule)
    { Attr.Style |= SS_RIGHT; }
  };  // class TRightStatic
//****************************************
class TNumEdit : public TEdit
  {  // an editor that allows only
     // '0'..'9', '-', and '.'
public:
   TNumEdit(PTWindowsObject AParent,
            int AnId, LPSTR AText,
            int X, int Y, int W, int H,
            WORD ATextLen, BOOL Multiline,
            PTModule AModule = NULL)
    : TEdit(AParent, AnId, AText, X, Y,
            W, H, ATextLen, Multiline,
            AModule)
    { }

  virtual void WMChar(RTMessage Msg) =
                     [WM_FIRST + WM_CHAR];
  };  // class TMaskedEdit

void TNumEdit::WMChar(RTMessage Msg)
  {  // only allow input if numeric
  char ch = Msg.WP.Lo;
  if ( (ch >= '0') && (ch <= '9')
    || ch == '-' || ch == '.' || ch < ' ')
    DefWndProc(Msg);
  else
    {
    MessageBeep(0);  // Bad Input!!
    // set Window Proc return value
    Msg.Result = 0;
    }
  }  // TNumEdit::WMChar()
//****************************************
const WORD MAX_TEXTLEN = 35;

class TMaskedEdit : public TEdit
  {
public:
   TMaskedEdit(PTWindowsObject AParent,
               int AnId, LPSTR AText,
               int X, int Y, int W, int H,
               WORD ATextLen,
               PTModule AModule = NULL)
    : TEdit(AParent, AnId, AText,
            X, Y, W, H, ATextLen,
            FALSE, AModule)
    { Attr.Style |= ES_MULTILINE; }

  virtual void WMGetDLGCode(RTMessage Msg)
      = [WM_FIRST + WM_GETDLGCODE];
  virtual void WMChar(RTMessage Msg)
      = [WM_FIRST + WM CHAR];

  // filtering function to be redefined
  // in descendants
  // - translate ch if necessary
  // - return TRUE if okay to insert char
  virtual int Filter(BYTE &ch, LPSTR Text,
                     int SelBeg,
                     int SelEnd)
    { return TRUE; }
  };  // class TMaskedEdit

void TMaskedEdit::WMGetDLGCode
                         (RTMessage Msg)
  {// modify to respond that we don't
   // want to process VK_TAB ourselves.
  DefWndProc(Msg);
  Msg.Result &=
     ~(DLGC_WANTTAB | DLGC_WANTALLKEYS);
  }  // TMaskEdit::WMGetDLGCode()

void TMaskedEdit::WMChar(RTMessage Msg)
  {
  // get current state of edit
  int BegSel, EndSel;
  GetSelection(BegSel, EndSel);
  char Text[MAX_TEXTLEN] ;
  GetText(Text, sizeof(Text));
  // pass backspace, etc.
  if ( (Msg.WP.Lo < ' ') ||
   Filter(Msg.WP.Lo,Text,BegSel,EndSel))
    DefWndProc(Msg); // dflt, insert char
  else
    {
    MessageBeep(0);  // Bad Input!!
    // set Window Proc return value
    Msg.Result = 0;
    }
  }  // TMaskedEdit::WMChar()
//****************************************
class TMNumEdit : public TMaskedEdit
  {  // a smarter NumEdit
    // only allows '-' in first position
    // if '.' is typed, deletes all text
    // between current selection and
    // decimal point
public:
   TMNumEdit(PTWindowsObject AParent,
             int AnId, LPSTR AText,
             int X, int Y, int W, int H,
             WORD ATextLen,
             PTModule AModule = NULL)
    : TMaskedEdit(AParent, AnId, AText,
                  X, Y, W, H,
                  ATextLen, AModule)
    { Attr.Style |= ES_RIGHT; }

  virtual int Filter(BYTE &ch,LPSTR Text,
                     int SelBeg,
                     int SelEnd);
  };  // class TMNumEdit

int TMNumEdit::Filter
               (BYTE &ch, LPSTR Text,
                int SelBeg, int SelEnd)
  {// determine if ch can be inserted
   // '-' only allowed at start of field
  if ( (ch == '-') && (SelBeg == 0) )
    {
    if (Text[SelEnd] == '-')
      SetSelection(SelBeg, SelEnd+1);
    return TRUE;
    }
  // look for existing decimal point
  LPSTR DecPtr = _fstrchr(Text,'.');
  int DecPos = LOWORD(DecPtr) -
                         LOWORD(Text);
  if (ch == '.')
    {
    if (!DecPtr)
      return TRUE;
    // 2nd decimal pt. requested
    if (DecPos < SelBeg)
      return FALSE; // beep if beyond d/p
    if (DecPos >= SelEnd)
      { // if current '.' is beyond
        // selection, extend selection
        // to include '.'. This will
        // ensure old '.' is removed
       SetSelection(SelBeg, DecPos+1);
       }
    return TRUE;
    }  // if ('.')
  // if not digit, don't allow
  if ( (ch < '0') || (ch > '9') )
    return FALSE;
  if ((DecPos != SelBeg) &&
                  (SelBeg == SelEnd))
    // simulate 'replace mode'
    // unless at dec.
    SetSelection(SelBeg, SelBeg+1);
  return TRUE;
  }  // TMNumEdit::Filter()
/*****************************************
// Normal Application Stuff
class TTestWindow : public TWindow
  {
public:
  TEdit *Editor[3];
  TTestWindow::TTestWindow
               (PTWindowsObject AParent,
                LPSTR ATitle);
  };  // class TTestWindow

TTestWindow::TTestWindow
               (PTWindowsObject AParent,
                LPSTR ATitle)
  : TWindow(AParent, ATitle)
  {
  new TRightStatic(this,-1,"Normal Edit:",
                   5, 10, 150, 25, 0);
  Editor[0] = new
    TEdit(this,-1,"235.54",160, 10,
          150, 25, MAX_TEXTLEN, FALSE);
  new TRightStatic(this,-1,
                   "Dumb Numeric:",
                   5, 40, 150, 25, 0);
  Editor[1] = new
    TNumEdit(this, -1, "235.54",160, 40,
             150, 25, MAX_TEXTLEN, FALSE);
  new TRightStatic(this, -1,
                   "Smart Numeric:",
                   5, 70, 150, 25, 0);
  Editor[2] = new
    TMNumEdit(this, -1, "235.54",160, 70,
              150, 25, MAX_TEXTLEN);
  AssignMenu("COMMANDS");
  EnableKBHandler();
  }  //  TTestWindow::TTestWindow()
//****************************************
class TTestApp : public TApplication
  {
public:
  TTestApp(LPSTR AName, HANDLE hInstance,
           HANDLE hPrevInstance,
           LPSTR lpCmdLine, int nCmdShow)
    : TApplication(AName, hInstance,
                   hPrevInstance,
                   lpCmdLine, nCmdShow)
    {};
  virtual void InitMainWindow();
  };  // class TTestApp

void TTestApp::InitMainWindow()
  { MainWindow = new
             TTestWindow(NULL, Name); }
//****************************************
int PASCAL WinMain(HANDLE hInstance,
                   HANDLE hPrevInstance,
  LPSTR lpCmdLine, int nCmdShow)
  {
  TTestApp TestApp("Masked Editor Tester",
    hInstance, hPrevInstance,
    lpCmdLine, nCmdShow);
  TestApp.Run();
  return TestApp.Status;
  }  // WinMain()
// end of MASKEDIT.CPP
```

Figure 2 - MASKEDIT.CPP, uses TNumEdit and TMaskedEdit

A WORLD OF

The widest range of software is only a phone call away

FAST - Federation Against Software Theft says £300m was lost by U.K software publishers alone through piracy in the year to March 1990. Software Paradise support and encourage the ethical use of software. To report copyright violations call FAST on (0628) 660377

The Widest Selection of Applications and Development Tools Anywhere

Place Orders Free on 0800-378-873

Just Call Us - It's That Easy!

Why struggle with traffic? Why fight crowds? Just pick up the phone and say hello to the convenient way to buy ALL your PC software. Software Paradise Expressline is the fast, new distribution system for all of today's popular titles. Choose from the widest range of software available from any single source, all listed in our FREE 100-page Buyers' Guide. Place your orders FREE and the software will be despatched to you directly.

Software Paradise has supplied 100,000's of customers since 1986 and has fast become the U.K's leading software supplier. Our service is both convenient and affordable with up to 50% off manufacturers RRP's and the widest range anywhere!

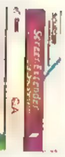
Call us today and discover a World of Software!

Software Paradise®

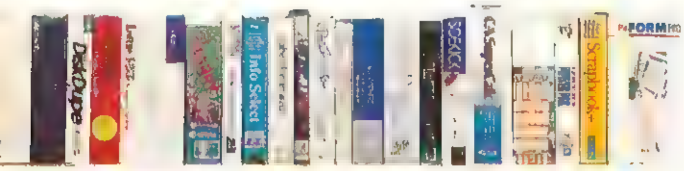
MORE SOFTWARE THAN ANYONE ELSE!

Call 0222 887521 • Fax 0222 862209

SOFTWARE *To Go!*



Over 15,000 Software



Products for

DOS, Windows™,

OS/2®, Mac

& UNIX®

* If you find an identical software product advertised in the national computer press by another mail-order company based in the U K at a legitimate, current price - tell us at the time of placing your order and we will happily BEAT that price and SAVE YOU MONEY! Join the 1000's of customers who have already benefited!

Don't forget - We'll BEAT Any Nationally Advertised Price! *

WHAT MAKES US DIFFERENT

- ✓ FREEPHONE 0800-378-873 Orders Only
- ✓ FREEPOST Mailing Address
- ✓ 24-Hour FAX Orderlines
- ✓ Up to 50% Discount On Over 15,000 Software Titles
- ✓ Pre-Sales Consultation
- ✓ FREE Post-Sales Technical Support
- ✓ Bulletin Board 0222 865551 (2400,8,N,1)
- ✓ Compuserve Mailbox 100015, 3570
- ✓ Unbeatable Prices *
- ✓ Next Day Delivery
- ✓ Huge Software Inventory
- ✓ FREE Literature Department
- ✓ Friendly & Knowledgeable Sales Staff
- ✓ VISA, ACCESS & Mastercard Accepted
- ✓ No Credit Card Charges Until Shipping
- ✓ Corporate P.O's Welcome
- ✓ Educational & Volume Discounts
- ✓ Latest Versions Guaranteed with Full Documentation & Manufacturers Warranties
- ✓ Over 6 Years Experience Providing the Very Best in U.K and U.S Software
- ✓ Stable & Secure Company with Impressive Track Record & Enviable Balance Sheet

Products shown in this advertisement represent only a small sample of our full range. Please specify Media Size when ordering. Goods are not offered on a trial basis. All our prices exclude Carriage & VAT.

All Trademarks & Registered Trademarks are hereby acknowledged
© Copyright '91/92 Software Paradise All Rights Reserved
Software Paradise, Avenue House, King Edward Avenue, Caerphilly, Mid Glamorgan, CF8 IHF

FREEPOST -TODAY-

NO STAMP NEEDED

□ Please rush me my **FREE** copy of *'The Discerning Persons Guide to Software'*

NAME

COMPANY

ADDRESS

TEL FAX

Mail to: Software Paradise, FREEPOST, Caerphilly CF8 1ZZ
.EXE 4/92

CIRCLE NO. 849

WMChar first learns the contents and selection location of the Edit field, then calls Filter(), sending it the input character, current string, and selection. If Filter() returns TRUE, DefWndProc() is called to insert the newly typed character. If Filter() returns FALSE, the bell is sounded, and no characters are inserted. As TMaskedEdit::Filter() always returns TRUE, a stock TMaskedEdit will allow anything.

One disconcerting feature of TMaskedEdit::WMChar() is that it makes a copy of the entire edit text every time a key is pressed. Although sending the Windows API message EM_GETHANDLE to an Edit window is supposed to get you a LOCALHANDLE to the buffer containing the text (eliminating the need to make a copy), this only works if the Edit is inside a dialog box, and if the Edit is created with ES_LOCALEDIT style. I tried setting ES_LOCALEDIT for my TEdit windows, and met with total failure - I suppose because they are children of normal windows rather than dialog boxes.

By setting a breakpoint on Edit windows' WndProc() and tracing through, I did learn that the first two extra bytes of any Edit window's 'Window Struct' are a near pointer to a descriptor table for that window. I even figured out enough about the table's contents to write the program using that information. But as this is undocumented (by Microsoft, that is) I don't want to take the responsibility of putting it into the hands of the unwitting. In the end I replaced it with documented calls, guaranteed to work. On my 386SX, at least, copying a string once each time a key is pressed doesn't seem to slow things down noticeably.

If you're really curious about this descriptor, use GetWindowWord(HWindow, 0) to get the value, then use it as a near pointer (hint: the first word of the structure is a char** which gives the address of the string). And don't blame me when your program doesn't work with Windows 3.1!

Right Justification

In any program I write, I always seem to get into some situation which creates a never ending downward spiral, eventually auguring me into the carpet. In this case, my problem was right justification - columns of numeric edit fields look lots nicer when they're right justified.

Step one - set the ES_RIGHT bit in the TEdit's Attr member. This changed nothing.

Step Two - learn from a colleague that ES_RIGHT only takes effect in Edits which also have ES_MULTILINE style. Step Two B - notice that TEdits have an argument to their constructor called 'Multiline', set Multiline to TRUE. OWL very kindly adds vertical and horizontal scroll bars to my teeny tiny window. Eeek!

Step Three - rather than using the Multiline parameter of TEdit's constructor, I call TEdit with Multiline FALSE, then set the ES_MULTILINE bit in the window's Attr during TMaskedEdit's constructor. This does the trick; I avoid OWL's default processing for Multline, but as the Windows API CreateWindow() call isn't actually made until later, Windows still gets the ES_MULTILINE flag. Now when I set ES_RIGHT, the window really does right justify the text. I enjoy a few minutes of euphoria before I happen to press the TAB key and notice that, for all other TEdits, TAB moves focus to the next TEdit, but once focus gets to a TMaskedEdit, it is stuck.

```
// check for Tab and backTab and
// change focus accordingly
if (Msg.WP.Lo == VK_TAB)
   {
   PTWindow NewFocus = this;
   if (GetKeyState(VK_SHIFT) < 0)
      do
      { NewFocus =
          (PTWindow)NewFocus->Previous();
      }
      while (
       !(NewFocus->Attr.Style & WS_TABSTOP)
              && (NewFocus != this) );
   else
      do
      { NewFocus =
          (PTWindow)NewFocus->Next();
      }
      while (
       !(NewFocus->Attr.Style & WS_TABSTOP)
              && (NewFocus != this) );
   SetFocus(NewFocus->HWindow);
   Msg.Result = 0;
   return;
   }   // if (VK_TAB)
```

Figure 3 - Abortive attempt at TAB processing

Tab Processing

It happens that, for single line Edits, TAB is intercepted by Windows and used to switch focus to the next sibling window with WS_TABSTOP style. But if the Edit is ES_MULTILINE, TAB is passed directly to the Edit window. If you want TAB to switch focus from an ES_MULTILINE Edit, you have to process it yourself.

As TAB is sent to the Window with the WM_CHAR message, I tried processing it with the addition to TMaskedEdit::WMChar() shown in Figure 3. Note that I had to use the Windows API directly to learn if SHIFT was pushed. Although processing the TAB key in this manner worked, there were (as usual) problems.

The first problem is that Windows' default TAB processing automatically does a 'SelectAll' of the newly focused window, while my own TAB processing doesn't. As I can't be sure that the new window is TEdit, I can't safely call SetSelection() (which isn't defined in TEdit). The only possibility would be to use the Windows API to send an EM_SETSEL message to the window, but this could also lead to disaster if another (non-Edit) window interpreted that command differently (EM_SETSEL is just a number, and only defined for Edits).

The second trouble with my TAB processing concerns dangerous type-casting. Although Next() and Previous() return TWindowsObject*, I am casting the return to TWindow* so I can access TWindow::Attr. If a non-TWindow descendant of TWindowsObject (eg TDialog) was in the window chain, I could be accessing a member that doesn't exist! And unfortunately, the IsA() member of all TWindowsObjects only tells me TRUE/FALSE if the object in question is *exactly* the same class as I requested; there is no way of learning if the object is a descendant of the questioned class.

Because of these two problems, this method of TAB processing was unacceptable, so I went back to the Windows API Function Reference, where I came across an obscure message called WM_GETDLGCODE. This message returns a bit field that tells some of the capabilities of the queried window. In particular, if the DLGC_WANTTAB and DLGC_WANTALLKEYS bits are set, it means that the window does its own TAB processing.

To make Windows process TAB for TMaskedEdits, we simply define a WMGetDLGCode() function which gets the original bit field and resets DLGC_WANTTAB and DLGC_WANTALLKEYS before returning the result. Now the TAB is intercepted by Windows, and TMaskedEdits never see it. As Windows is doing this processing, it takes care of searching for WS_TABSTOPs, SetSelection(), and the like. The result is exactly as I wanted. I now have a single line Edit which can be right justified, but otherwise behaves just like a normal single line Edit.

A Useful TMaskedEdit

As with most virtual functions, Filter() only gives TMaskedEdit potential. To have capability, you have to derive a new class. A quick look at TMNumEdit will show why it is important to know the string contents and location of the selection. For example, if a minus ('-') is typed you must make sure that 1) you are positioned at the beginning of the string, and 2) there is no other '-' character in the string.

TMNumEdit::Filter() uses the information sent to it to do some fairly intelligent processing. For example, while characters typed just before '.' are inserted, those typed anywhere else are entered in 'replace' mode. Also, typing '.' when the selection is left of an existing decimal point deletes all numbers in between. Notice that Filter doesn't do the actual deletion itself, it simply extends the selection to enclose those characters it wants deleted; all selected text is deleted prior to any insertion. Replace mode is simulated by selecting one character ahead of the caret,. causing it to be deleted before the new character is inserted.

The example program in Figure 2 creates Edits of all three types so that you can compare their features.

Dialog Boxes

Now we have a wonderful Edit class. The next step is to use it in a Dialog Box designed with Resource Workshop, right? Sorry, controls in a dialog box must be of a unique Windows registration class. In all the modifications we've made to TMaskedEdit, it still uses the standard Windows 'Edit' class. To make a TMaskedEdit that can be called from a dialog box, you have to kick back to plain C and learn something about Windows Custom Controls. If you're interested in that, take a look at the .DOC files included with BC++ v3.0.

And send me a copy of what you write!

Usually willing to take on any task with enthusiasm, Laine Stump is too tired to write anything interesting about himself this month.

You can contact Laine via modem at the PC Tech BBS (0101-612-345-4656, evenings US time) or by mail at: Bilkent University, Lojmanlari 3/9, 06533 Bilkent / Ankara, TURKEY.

Lazy typists may obtain this code by sending in a disk as per the rules laid out on Page 1, column 1. Please mark your envelopes 'TMASKEDEDIT'.

Heaps of XMS for Turbo Pascal

Fed up with being unable to use all that lovely XMS memory? Julian Bucknall was as well, so he wrote an XMS heap manager.

I'd had a specific problem, and I'd worked out an answer. I was feeling pretty damn good.

The problem was that my program needed to calculate a single value when the user pressed a button. Display It As Fast As Possible, said the voice with the salary cheque. The single value depended on a whole set of partial calculations that in turn were derived from a set of 20 or so input values. Calculating the single result from scratch took an appreciable amount of time, even when I'd done the usual code efficiency improvement exercises. The brain wave came, store the partial calculations in memory all the time. The size of the record structure of these partial values made me blanch, 30 KB, but so what? I coded it in. That's when I felt pretty damn good.

Then the voice spoke again: We Need A Set Of These Values, All Derived From Different Input. All Displayed As Fast As Possible.

I was stuck, I didn't have enough heap memory left to store multiple sets of these partial result structures. Saving and retrieving from disk would be too slow; it would be faster to recalculate from square one. So it was either EMS or XMS time. My users had from 4 MB to 12 MB RAM, all had QEMM; there was plenty of RAM out there. I was already using EMS for the program overlay and I didn't want to get stuck debugging the EMS map switching process. XMS it was then. I set to and designed and wrote an XMS heap manager as a Turbo Pascal unit.

Design criteria

My main criteria were these: the manager was not to allocate all of XMS memory for its purposes, but was to get it in pages of some defined size; there must be routines to allocate and deallocate XMS memory blocks (obviously!), and to get data in and out of XMS; there must be a garbage collection routine to compress out deallocated blocks; I wanted to refer to XMS data blocks via some kind of handle scheme; whenever I copied data to and from some XMS block I didn't want to have to keep passing the data size (I am a lazy programmer; I am also apt to code these values wrongly on the morning after the night before).

Basics first. The manager gets XMS memory in pages of 256 KB. The application blocks allocated within each page will be a multiple of 16 bytes in size (ie one paragraph). Each application block will be preceded by a block header which will detail things like the paragraph size of the block, the size of the user data, the paragraph offset of the next application block, and so on. This kind of scheme is used by DOS itself to track normal memory. The minimum allocation of memory from a page will be two paragraphs, and hence the maximum number of blocks in a page will be 8192. Each application block allocated would be referred to by the paragraph offset of its header, the data in the block would be found one paragraph further on. Blocks would be allocated from the lowest offset upwards.

The handle that the application was going to use would obviously have to point to a page and a paragraph offset within that page. However, remember that I wanted the heap manager to be able to compress the heap. This would result in application

$80	Function not implemented
$81	Vdisk was detected
$82	An A20 error occurred
$8E	A general driver error
$8F	Unrecoverable driver error
$90	HMA does not exist
$91	HMA is already in use
$92	DX is less than the /HMAMIN= parameter
$93	HMA is not allocated
$94	A20 line still enabled
$A0	All extended memory is allocated
$A1	All available extended memory handles are allocated
$A2	Invalid handle
$A3	Source handle is invalid
$A4	Source offset is invalid
$A5	Destination handle is invalid
$A6	Destination offset is invalid
$A7	Length is invalid
$A8	Move has an invalid overlap
$A9	Parity error occurred
$AA	Block is not locked
$AB	Block is locked
$AC	Block lock count overflowed
$AD	Lock failed
$B0	Only a smaller UMB is available
$B1	No UMB's are available
$B2	UMB segment number is invalid

Figure 1 - XMS driver error codes

blocks being moved around, and so the block handle could not contain the actual offset. The reason is that there would be no way for the XMS heap manager to update all handle variables within the program's data, stack and heap space. Turbo Pascal's own heap manager has the same problem, which is why the normal heap is not compressible. Thus there must be some level of indirection. I decided that the handle would have an index to an internal table of page offsets. To get the actual paragraph offset, extract the index from the handle, and the entry in the table referenced by that index would point to the data block header. The compression routine would be able to juggle around the offsets within the table as it performed the page compaction. To minimise the amount of normal memory used, the offset table would reside in the first 16 KB of the page.

How it works

To describe the workings of the XMS heap manager, it will be easier to describe what happens during normal program execution, rather than a bald description of each routine. Please refer freely to the listing of the unit in Figure 2.

The first thing that happens is that the XMS heap manager is initialised. The initialisation routine `InitXMS` checks that the XMS driver is present and gets the driver's entry point address. It initialises various other variables, including its internal list of XMS pages `XMSMap`. It installs an exit routine `ReleaseXMS` into Turbo Pascal's exit procedure chain so that it can clean up by releasing any allocated XMS on program termination. It does not at this stage get any memory from the XMS driver.

A quick note is required here. For some unknown reason (to me anyway) programs do not communicate with the XMS driver via an interrupt, as with the EMS driver. Instead the program obtains an address from the XMS driver, and does a `FAR CALL` to it instead. There are only two interrupt calls to the XMS driver, the first is an 'are you there?' enquiry, the second is a 'give me your entry point address' request. Hence you have to bite the bullet and start writing in assembler; I have chosen to use the inbuilt assembler in Turbo Pascal 6.0 for ease of programming (please note that you will have to code these as external assembler routines for Turbo Pascal 5.5 and earlier). Another restriction is that data is written to and from XMS in blocks of even size; no odd-sized blocks are allowed, their size must be rounded up to the nearest even number. The heap manager takes care of this restriction, you do not have to consciously create even-sized variables for use with XMS.

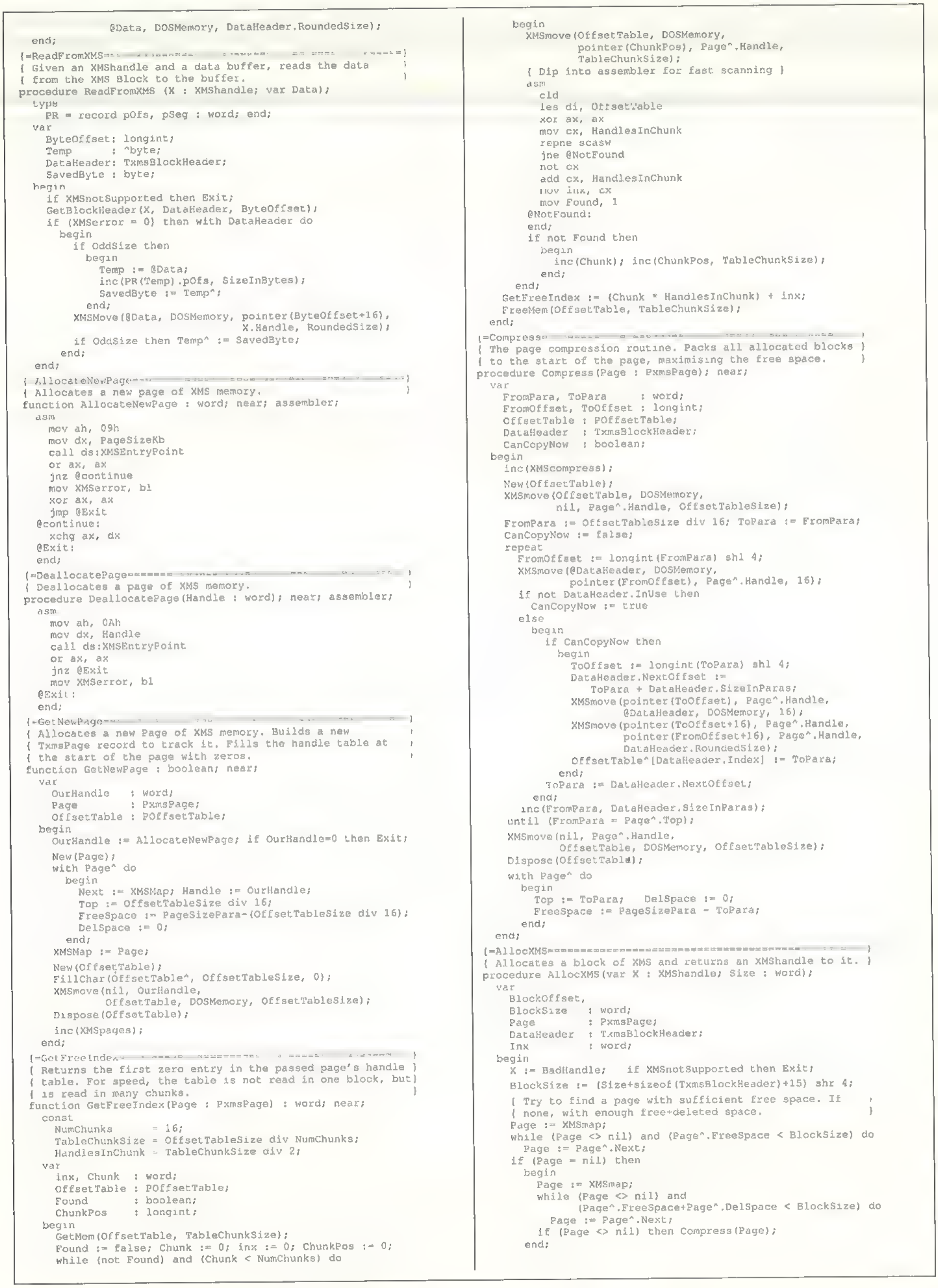
At some point in your program, you want to allocate an XMS block and hence call `AllocXMS`, passing an `XMShandle` variable and the size of block required. `AllocXMS` rounds this up to the nearest paragraph size, including the block header. It now tries to find an already allocated page with enough free space to accommodate this request. Details about allocated

```
UNIT XMSheap;
{$A+,B-,E+,F+,G+,I-,N+,O-,R-,S-,V-,X+}
INTERFACE
const
  XMSpresent : boolean = false; { True if XMS is present }
  XMSerror   : byte = 0;        { XMS error number  }
                                { Debugging counters... }
  XMSpages    : word = 0;       {    - num of pages     }
  XMSallocs   : word = 0;       {    - allocations      }
  XMSdeallocs : word = 0;       {    - deallocations    }
  XMScompress : word = 0;       {    - compressions     }
type
  XMShandle = record   { Definition of an XMShandle        }
    Index  : word;     { ...Index into page's offset table }
    Handle : word;     { ...Actual handle of page          }
  end;
procedure InitXMS;
procedure AllocXMS(var X : XMShandle; Size  : word);
procedure DeAllocXMS(var X : XMShandle);
procedure WriteToXMS (X : XMShandle; var Data);
procedure ReadFromXMS(X : XMShandle; var Data);
IMPLEMENTATION
const
  SigConst = $424A; { Data Block header signature }
  PageSizeKb   = 256;                { Page size in Kb    }
  PageSizePara = PageSizeKb * 64;    { Page size in paras }
  MaxBlocks    = PageSizePara div 2; { Max blocks/page    }
  OffsetTableSize = MaxBlocks * 2;   { Offset table size  }
  BadHandle : XMShandle = (Index : $FFFF; Handle : $0);
                        { Value to denote a bad XMShandle }
  DOSMemory = 0;  { XMS driver's handle for normal memory }
type
  TxmsBlockHeader = record   { The header for an XMS Block }
    Signature   : word;      { Always $424A                }
    SizeInParas : word;      { Size of XMS block incl hdr  }
    NextOffset  : word;      { Address of next block       }
    SizeInBytes : word;      { Size of user data in bytes  }
    RoundedSize : word;      { Size rounded to even number }
    OddSize     : boolean;   { True if Odd user size       }
    InUse       : boolean;   { True if not deallocated     }
    Index       : word;      { Index in pages offset table }
    Filler      : word;      { Force header to para size   }
  end;
  POffsetTable = ^TOffsetTable;  { The page's offset table }
  TOffsetTable = array [0..pred(MaxBlocks)] of word;
  PxmsPage = ^TxmsPage;
  TxmsPage = record        { Definition of a page           }
    Next      : PxmsPage;  { Next item in the Map, or nil   }
    Handle    : word;      { Main XMS handle for this page  }
    Top       : word;      { Top of allocated space         }
    FreeSpace: word;       { Size of free space             }
    DelSpace : word;       { Total deleted space            }
  end;
```

```
var
  XMSEntryPoint : pointer; { XMS driver's entry point   }
  XMSMap : PxmsPage;       { Map of the Pages allocated }
  ExitSave : pointer;

{=XMSnotSupported========================================}
{ Returns true if no XMS, false otherwise. Sets XMSerror.  }
function XMSnotSupported : boolean; near;
  begin
    if XMSpresent then XMSerror := 0 else XMSerror := $80;
    XMSnotSupported := not XMSpresent;
  end;
{=XMSmove================================================}
{ Moves data from XMS to normal memory.                    }
procedure XMSmove(ToPtr   : pointer; ToHandle   : word;
                  FromPtr : pointer; FromHandle : word;
                  Size  : longint); near; assembler;
  asm
    mov dx, ds;             { Save Turbo's ds }
    mov ax, ss; mov ds, ax  { Set ds equal to ss }
    lea si, Size            { Get the address of Size }
    mov es, dx              { Set es to Turbo's ds }
    mov ah, 0Bh             { Call XMS move routine }
    call es:XMSEntryPoint
    mov ds, dx              { Restore our ds }
    or ax, ax               { Failure? }
    jnz @Exit              { Nope... continue }
    mov XMSerror, bl        { Yup, so tell user so }
  @Exit:
  end;
{=GetBlockHeader=========================================}
{ Given an XMShandle, returns its block header & offset.   }
procedure GetBlockHeader(X : XMShandle;
                         var DataHeader : TxmsBlockHeader;
                         var Offset : longint); near;
  var
    ParaOffset : word;
  begin
    { Get offset of block header in page }
    XMSmove(@ParaOffset, DOSMemory,
            pointer(X.Index * 2), X.Handle, 2);
    Offset := longint(ParaOffset) shl 4;
    { Get block header itself }
    XMSMove(@DataHeader, DOSMemory,
            pointer(Offset), X.Handle, 16);
  end;
{=WriteToXMS=============================================}
{ Given an XMShandle and a data buffer, writes the data to }
{ the XMS Block.                                           }
procedure WriteToXMS (X : XMShandle; var Data);
  var
    ByteOffset : longint;
    DataHeader : TxmsBlockHeader;
  begin
    if XMSnotSupported then Exit;
    GetBlockHeader(X, DataHeader, ByteOffset);
    if (XMSerror = 0) then
      XMSMove(pointer(ByteOffset+16), X.Handle,
```

Figure 2 - XMS heap manager unit



```
                  @Data, DOSMemory, DataHeader.RoundedSize);
  end;
{=ReadFromXMS==================================================}
{ Given an XMShandle and a data buffer, reads the data         }
{ from the XMS Block to the buffer.                            }
procedure ReadFromXMS (X : XMShandle; var Data);
  type
    PR = record pOfs, pSeg : word; end;
  var
    ByteOffset: longint;
    Temp      : ^byte;
    DataHeader: TxmsBlockHeader;
    SavedByte : byte;
  begin
    if XMSnotSupported then Exit;
    GetBlockHeader(X, DataHeader, ByteOffset);
    if (XMSerror = 0) then with DataHeader do
      begin
        if OddSize then
          begin
            Temp := @Data;
            inc(PR(Temp).pOfs, SizeInBytes);
            SavedByte := Temp^;
          end;
        XMSMove(@Data, DOSMemory, pointer(ByteOffset+16),
                                  X.Handle, RoundedSize);
        if OddSize then Temp^ := SavedByte;
      end;
  end;
{=AllocateNewPage==============================================}
{ Allocates a new page of XMS memory.                          }
function AllocateNewPage : word; near; assembler;
  asm
    mov ah, 09h
    mov dx, PageSizeKb
    call ds:XMSEntryPoint
    or ax, ax
    jnz @continue
    mov XMSerror, bl
    xor ax, ax
    jmp @Exit
  @continue:
    xchg ax, dx
  @Exit:
  end;
{=DeallocatePage===============================================}
{ Deallocates a page of XMS memory.                            }
procedure DeallocatePage(Handle : word); near; assembler;
  asm
    mov ah, 0Ah
    mov dx, Handle
    call ds:XMSEntryPoint
    or ax, ax
    jnz @Exit
    mov XMSerror, bl
  @Exit:
  end;
{=GetNewPage===================================================}
{ Allocates a new Page of XMS memory. Builds a new             }
{ TxmsPage record to track it. Fills the handle table at       }
{ the start of the page with zeros.                            }
function GetNewPage : boolean; near;
  var
    OurHandle   : word;
    Page        : PxmsPage;
    OffsetTable : POffsetTable;
  begin
    OurHandle := AllocateNewPage; if OurHandle=0 then Exit;
    New(Page);
    with Page^ do
      begin
        Next := XMSMap; Handle := OurHandle;
        Top := OffsetTableSize div 16;
        FreeSpace := PageSizePara-(OffsetTableSize div 16);
        DelSpace := 0;
      end;
    XMSMap := Page;
    New(OffsetTable);
    FillChar(OffsetTable^, OffsetTableSize, 0);
    XMSmove(nil, OurHandle,
            OffsetTable, DOSMemory, OffsetTableSize);
    Dispose(OffsetTable);
    inc(XMSpages);
  end;
{=GetFreeIndex=================================================}
{ Returns the first zero entry in the passed page's handle }
{ table. For speed, the table is not read in one block, but}
{ is read in many chunks.                                  }
function GetFreeIndex(Page : PxmsPage) : word; near;
  const
    NumChunks      = 16;
    TableChunkSize = OffsetTableSize div NumChunks;
    HandlesInChunk = TableChunkSize div 2;
  var
    inx, Chunk  : word;
    OffsetTable : POffsetTable;
    Found       : boolean;
    ChunkPos    : longint;
  begin
    GetMem(OffsetTable, TableChunkSize);
    Found := false; Chunk := 0; inx := 0; ChunkPos := 0;
    while (not Found) and (Chunk < NumChunks) do
      begin
        XMSmove(OffsetTable, DOSMemory,
                pointer(ChunkPos), Page^.Handle,
                TableChunkSize);
        { Dip into assembler for fast scanning }
        asm
          cld
          les di, OffsetTable
          xor ax, ax
          mov cx, HandlesInChunk
          repne scasw
          jne @NotFound
          not cx
          add cx, HandlesInChunk
          mov inx, cx
          mov Found, 1
        @NotFound:
        end;
        if not Found then
          begin
            inc(Chunk); inc(ChunkPos, TableChunkSize);
          end;
      end;
    GetFreeIndex := (Chunk * HandlesInChunk) + inx;
    FreeMem(OffsetTable, TableChunkSize);
  end;
{=Compress=====================================================}
{ The page compression routine. Packs all allocated blocks }
{ to the start of the page, maximising the free space.     }
procedure Compress(Page : PxmsPage); near;
  var
    FromPara, ToPara     : word;
    FromOffset, ToOffset : longint;
    OffsetTable : POffsetTable;
    DataHeader  : TxmsBlockHeader;
    CanCopyNow  : boolean;
  begin
    inc(XMScompress);
    New(OffsetTable);
    XMSmove(OffsetTable, DOSMemory,
            nil, Page^.Handle, OffsetTableSize);
    FromPara := OffsetTableSize div 16; ToPara := FromPara;
    CanCopyNow := false;
    repeat
      FromOffset := longint(FromPara) shl 4;
      XMSmove(@DataHeader, DOSMemory,
              pointer(FromOffset), Page^.Handle, 16);
      if not DataHeader.InUse then
        CanCopyNow := true
      else
        begin
          if CanCopyNow then
            begin
              ToOffset := longint(ToPara) shl 4;
              DataHeader.NextOffset :=
                 ToPara + DataHeader.SizeInParas;
              XMSmove(pointer(ToOffset), Page^.Handle,
                      @DataHeader, DOSMemory, 16);
              XMSmove(pointer(ToOffset+16), Page^.Handle,
                      pointer(FromOffset+16), Page^.Handle,
                      DataHeader.RoundedSize);
              OffsetTable^[DataHeader.Index] := ToPara;
            end;
          ToPara := DataHeader.NextOffset;
        end;
      inc(FromPara, DataHeader.SizeInParas);
    until (FromPara = Page^.Top);
    XMSmove(nil, Page^.Handle,
            OffsetTable, DOSMemory, OffsetTableSize);
    Dispose(OffsetTable);
    with Page^ do
      begin
        Top := ToPara;   DelSpace := 0;
        FreeSpace := PageSizePara - ToPara;
      end;
  end;
{=AllocXMS=====================================================}
{ Allocates a block of XMS and returns an XMShandle to it. }
procedure AllocXMS(var X : XMShandle; Size : word);
  var
    BlockOffset,
    BlockSize   : word;
    Page        : PxmsPage;
    DataHeader  : TxmsBlockHeader;
    Inx         : word;
  begin
    X := BadHandle;   if XMSnotSupported then Exit;
    BlockSize := (Size+sizeof(TxmsBlockHeader)+15) shr 4;
    { Try to find a page with sufficient free space. If    }
    { none, with enough free+deleted space.                }
    Page := XMSmap;
    while (Page <> nil) and (Page^.FreeSpace < BlockSize) do
      Page := Page^.Next;
    if (Page = nil) then
      begin
        Page := XMSmap;
        while (Page <> nil) and
              (Page^.FreeSpace+Page^.DelSpace < BlockSize) do
          Page := Page^.Next;
        if (Page <> nil) then Compress(Page);
      end;
```

Figure 2 - XMS beap manager unit (Continued)

How much longer can you afford to wait?

Create Overlaid Programs-Fast.

BLINKER™, the world's first and fastest dynamic overlay linker, reduces your link time to seconds and reduces program memory requirements. Now you can use one linker for all your software projects.

One Linker, Many Languages.

BLINKER 2.0 links and automatically overlays DOS programs written in Microsoft® C, BASIC, Assembler, QuickBASIC™, Fortran, Pascal, Watcom™ C, Zortech C++, Clipper®, FORCE® and in Borland® C, C++, Assembler, and more.

Save Time and Memory.

BLINKER removes the need for overlay structures, simplifies program design and reduces memory requirements to save you time, effort and memory.

Memory Swap Function.

BLINKER is the ONLY linker to offer an integrated memory swap function, so you can run other large programs from within your program, with negligible memory overhead.

BLINKER™
High Performance Dynamic Overlay Linker

Don't Settle for Less.

Other major features include full CodeView® support, use of EMS/XMS at program run-time, and enhanced execution speed of overlaid code.

Time is Money.

BLINKER offers all this in a fraction of the time it takes to link with your current overlay linker. You know time is money, and link time is no exception.

Free Demo

To try our free demo on your own code and for more information contact our U.K. distributor:

Call: +44-81-994-4842 or
FAX: +44-81-994-3441

QBS Software Ltd.
10, Barley Mow Passage
London W4 4PH

BLINKER is available in 5.25" or 3.5" diskette format

Price £199
plus shipping & handling
VISA MasterCard

Blink*inc*
P.O. Box 7154
Richmond VA
23221

© 1991 BlinkInc. Blinker is a trademark of Assembler Software Manufacturers Inc

CIRCLE NO. 850

pages are stored in a linked list called XMSMap, where each item on the list is a record with structure TxmsPage. So, AllocXMS walks this list looking for a page with the required amount of free space. If it can't find a block of free space large enough, it traverses the same list again, this time looking for a page with enough free and deleted space to satisfy the request. If it finds such a page, the page is compressed to maximise the free space; if it doesn't, a new page gets allocated from the XMS driver.

So, eventually, AllocXMS will find a page with enough free space. The first unused index from the page's offset table is found by a simple sequential search. That table element is updated with the paragraph offset of the start of the page's free space. AllocXMS can now build the data block's header (a TxmsBlockHeader structure), and write it to XMS. Finally, the page details are updated to show the new offset start of free space, and its size. The handle variable is returned with the page handle and the offset table index.

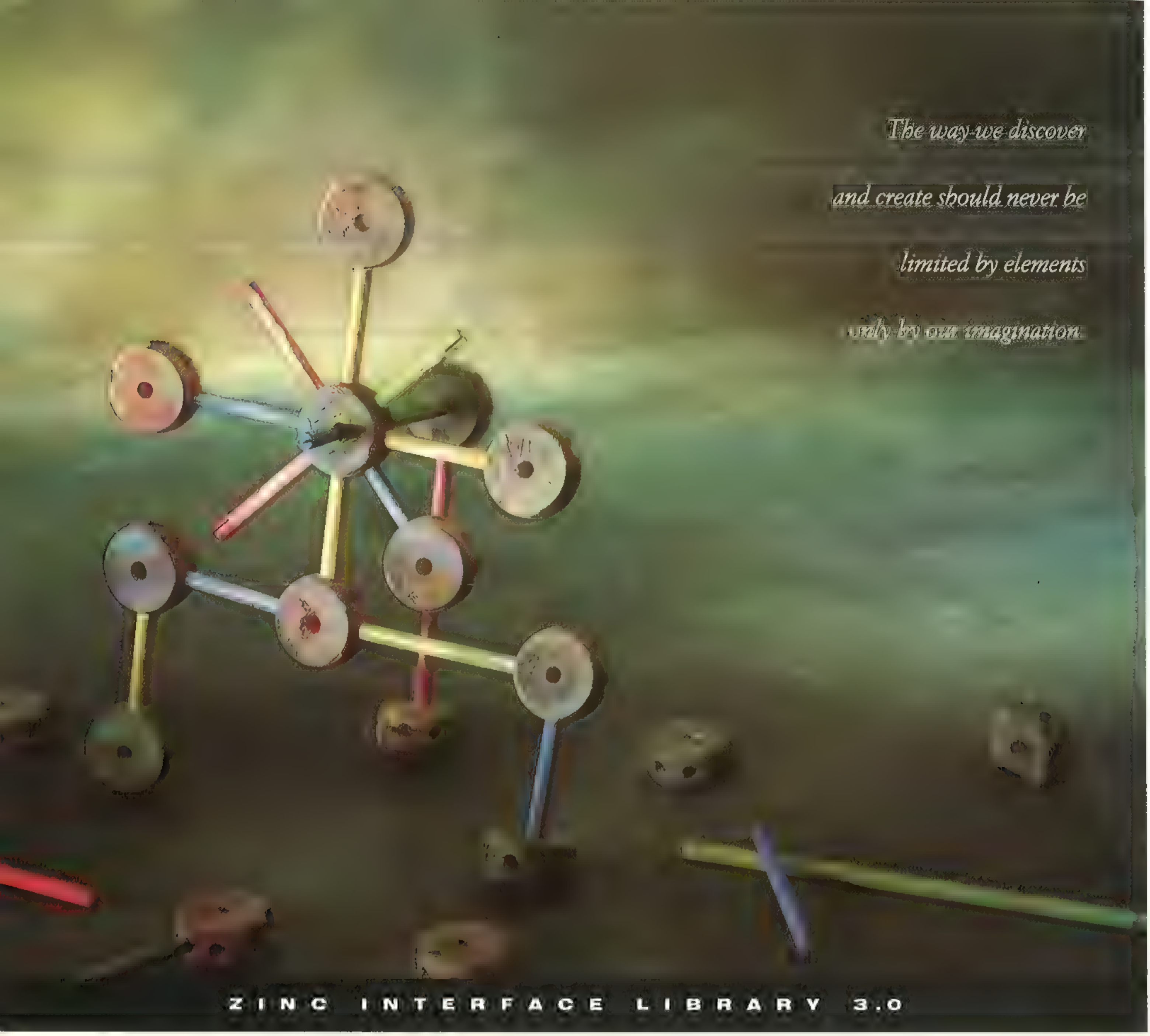
The program, having allocated its block, can now write some data to it using the WriteToXMS procedure. This procedure takes an XMShandle, and an untyped variable containing the data. WriteToXMS dereferences the handle, finds the paragraph offset of the data block in the page, and reads the data block header. This gives (among other things) the size of the data to be transferred, and that amount of data is copied from the untyped variable to XMS, immediately after the data block header. If the data size as originally defined was odd, one more byte than is required is transferred (XMS requires even-sized blocks remember), but this does no harm.

Compression could take an appreciable time but at least it is only done as a last resort

After a little while, the program will want to read from an XMS block. This is done by the ReadFromXMS procedure, and it is simply a reversal of the WriteToXMS processing above. The program must pass a variable that is large enough to hold the data passed back from XMS, otherwise memory will be overwritten! There is one slight gotcha though. What if the data size was originally odd? The heap manager must be careful not to overwrite memory. The technique used by the manager is to save the byte immediately after the data variable in normal memory, copy the even-sized data from the XMS data block (which will overwrite this byte), and then restore the original byte. This works well in practice, but could fail if the byte juggled with was required by an asynchronous process, for example.

```
      { Now, if no page was found with enough room, allocate }
      { ourselves another page from the XMS driver. We shall }
      { use index 0 as the handle table is blank.            }
      if (Page = nil) then
        begin
          if not GetNewPage then Exit;
          Page := XMSMap;    Inx := 0;
        end
      { Otherwise get a blank entry from our existing page's }
      { handle table }
      else
        Inx := GetFreeIndex(Page);
      { Update the page's handle table }
      XMSmove(pointer(Inx*2), Page^.Handle,
              @Page^.Top, DOSMemory, 2);
      { Set up the user's XMShandle }
      X.Handle := Page^.Handle; X.Index :  Inx;
      { Set up the block header, and write it }
      with DataHeader do
        begin
          Signature := SigConst;
          SizeInParas := BlockSize;
          NextOffset  := Page^.Top + BlockSize;
          SizeInBytes := Size;
          RoundedSize := Size;
          if Odd(Size) then
            begin OddSize := true; inc(RoundedSize); end;
          InUse := true;
          Index := Inx;
          Filler := 0;
        end;
      XMSMove(pointer(longint(Page^.Top) shl 4), Page^.Handle,
              @DataHeader, DOSMemory, 16);
      { Move Page^.Top to the bottom of free space again }
      Page^.Top := DataHeader.NextOffset;
      dec(Page^.FreeSpace, DataHeader.SizeInParas);
      inc(XMSallocs);
    end;
{=DeAllocXMS================================================}
{ Deallocates a block of XMS memory.                        }
procedure DeAllocXMS(var X : XMShandle);
  var
    ParaOffset : word;
    ByteOffset : longint;
    Page       : PxmsPage;
    DataHeader : TxmsBlockHeader;
  begin
    if XMSnotSupported then Exit;
    { Read the block header, mark it as deleted. }
    GetBlockHeader(X, DataHeader, ByteOffset);
    if (XMSerror <> 0) then Exit;
    DataHeader.InUse := false; DataHeader.Index := $FFFF;
    XMSMove(pointer(ByteOffset), X.Handle,
            @DataHeader, DOSMemory, 16);
    { Update the page's handle table }
    ParaOffset := 0;
    XMSmove(pointer(X.Index * 2), X.Handle,
            @ParaOffset, DOSMemory, 2);
    { Update the amount of free/deleted space in the page }
    Page := XMSmap;
    while (Page <> nil) and (Page^.Handle <> X.Handle) do
      Page := Page^.Next;
    if (Page <> nil) then
      if (Page^.Top = DataHeader.NextOffset) then
        begin
          inc(Page^.FreeSpace, DataHeader.SizeInParas);
          Page^.Top := DataHeader.NextOffset;
        end
      else
        inc(Page^.DelSpace, DataHeader.SizeInParas);
    { Make sure the user's handle is bad }
    X := BadHandle;
    inc(XMSdeallocs);
  end;
{=ReleaseXMS================================================}
{ Exit procedure for this unit. Releases any XMS allocated.}
procedure ReleaseXMS; far;
  var
    Page : PxmsPage;
  begin
    ExitProc := ExitSave;   if not XMSpresent then Exit;
    Page := XMSMap;
    while (Page <> nil) do
      begin
        DeallocatePage(Page^.Handle);
        XMSmap := Page^.Next; Dispose(Page); Page := XMSmap;
      end;
    XMSmap := nil;
  end;
{=InitXMS===================================================}
{ Determines whether XMS is present. If so initialises the }
{ Driver entry point variable, and various flags.          }
procedure InitXMS;
  label NoXMS;
  begin
    asm
      mov ax, 4300h      { Standard test for XMS driver }
      int 2Fh
      sub al, 80h        { al=80h => XMS present }
      jnz NoXMS          { No XMS, jump out of asm block }
      mov ax, 4310h      { Get driver entry point address }
      int 2Fh
      mov XMSEntryPoint.Word[0], bx
      mov XMSEntryPoint.Word[2], es
    end;
    XMSpresent := true;   XMSmap := nil;
    ExitSave := ExitProc; ExitProc := @ReleaseXMS;
   NoXMS:
   end;
end.
```

Figure 2 - XMS heap manager unit (Continued)


The way we discover
and create should never be
limited by elements
only by our imagination.
ZINC INTERFACE LIBRARY 3.0

PC Security from Microft Technology.

MENUGEN

MENUGEN provides a password protected menu interface. The system manager has complete control of what each user is permitted to do. Access to the DOS prompt can be prevented. MENUGEN has no memory overhead. **£48** + **VAT.**

CLAM

CLAM provides TOTAL security. It incorporates MENUGEN and D-LOCK. In addition it provides a number of other facilities, the most important of which is file encryption – the only way to provide a high level of security on a micro. **£148** + **VAT.**

D-LOCK

D-LOCK prevents access to the hard disk after booting from diskette. **£38** + **VAT.**

S-LOCK

S-LOCK clears the screen and locks the keyboard after a specified period of inactivity or on request. The screen is restored and the keyboard unlocked only when the correct password is input. **£38** + **VAT.**

Security Consultancy.

A professional computer security consultant can help you determine the steps required to protect your computer information and systems. The options offered range from a fixed price risk analysis with action advice to a full security audit.

Microft Technology Limited
The Old Powerhouse, Kew Gardens Station
Kew, Surrey TW9 3PS.
Telephone: 081-948 8255

Anti-virus software is also available. Corporate licences are available for all.products.

CIRCLE NO. 852

To deallocate the XMS data block procedure, `DeallocXMS` is called with the handle of the block to be deallocated. The block's header is read from XMS, is marked as deleted, and is rewritten. The block is then checked to see whether it is just underneath the page's free space, in which case the free space is incremented by the size of the block, and the free space start offset is moved down. If it occurs in the middle of the allocated space, then the deleted space counter is incremented.

The only process not yet described is the compression routine. This will get called only when a data block is required to be allocated, and the current page has enough free and deleted space to satisfy the request. The routine walks through the data block headers within the page looking for deleted blocks. When one is found, the next undeleted block is moved down to overwrite it, the block links are updated, and the offset table is updated to reflect the block's new offset. This process continues until there are no more deleted blocks to compress out, and the routine updates the free space variables. As you can appreciate, this process could take a considerable time, but at least it is only done as a last resort.

As far as errors go, I have defined a single error variable `XMSerror`. It is normally zero, but if an XMS operation fails, it will be set to the error code returned by the XMS driver. The majority of these would be rebootable offences, implying some kind of driver memory overwrite. The list of error numbers and meanings is given in Figure 1. Also, at the moment, there is no code to check whether a passed handle is valid or not. In a pre-production program, it would be a good idea to write a debug routine to trap this kind of error.

Afterthoughts

The overall efficiency of the heap manager could be improved by minimising the amount of automatic compression done. Either don't compress until you have to: ie keep on getting more pages from the XMS driver until all XMS memory has been used up. Otherwise, find the page with the least number of blocks allocated that has enough free and deleted space, and compress that instead. Another strategy to improve efficiency would be to recode various routines in assembler, using a profiler to find the ones used most of all.

The more observant of you will have noticed that the `XMSheap` unit has been defined as non-overlayable. There is nothing in the code of the unit that makes such a restriction applicable, however I have done so because I wanted to use it to write an XMS driver for the overlay manager. That, however, is another story for another time.

EXE

Julian Bucknall has been designing and programming in a variety of languages for 12 years, but is getting to like this writing lark. A retraining to C++ is in the offing; so far he has resisted. He can be reached on CompuServe on [100063,145], mainly hanging out in the BPROGA forum.

Turbo Pascal 6.0 Professional is a product from Borland International (0734 321150), and is available from Grey Matter (0364 53499) at £137.

Lazy typists may obtain this code by sending in a disk as per the rules laid out on Page 1, column 1. Please mark your envelopes 'XMS'.

Inside Clipper

What with tokenised code, virtual memory management and multiple stacks, Clipper's internals present something of a challenge. Jud Cole sorts them out.

My aim in this article is to show how a Clipper application works internally, with a view to writing more compact and efficient Clipper applications. I will describe public, private, local and static variable classes, storage of memory variable values of different types, and how the dynamic paging system manages Clipper 5.0 code at application run-time.

Clipper's symbol table

The Clipper language, and the dBASE language on which Clipper was originally based, is a dynamic language with a number of very powerful constructs. These allow and cause certain functions normally performed by the compiler to be postponed until run-time, such as setting the type of variables and using macros to create new variables not known at compile time.

Because of this dynamic nature, at run-time Clipper requires more information about variables and procedures than traditional lower-level languages such as Pascal, C and Modula-2. Some of this information is available at compile and link time, such as the name of the variable, but some of it, such as its type, will only be available once the application has started executing.

For these reasons, each Clipper .OBJ file is created with a symbol table of 16 bytes per symbol, and all code in the .OBJ file refers to that symbol table. At run-time the symbol table entry is used to point to the control information and value or code for the symbol. The symbol table is created in its entirety in the root of the application, and can grow above 64 KB, so it can significantly affect the amount of conventional memory required by the application. This is why even 100% overlaid applications grow when code is added.

Clipper 5.0 introduced static and local variables to the language to encourage better and more efficient coding practices. Another important benefit is that these classes of variables do not require a symbol table entry as they cannot be accessed via macros. Changing as many public and private variables as possible to local or static variables can therefore significantly reduce the amount of conventional memory required.

The major linkers now available (Blinker and .RTLink) remove the duplicate symbols from the symbol tables in the various .OBJ and .LIB files at link time, creating one large consolidated symbol table. This process, known as symbol table compression, can significantly reduce the run-time memory requirement of the .EXE, leaving more memory for the application's data and overlays. All the duplicates are removed except the symbols belonging to procedures declared as static, since these are local to each .OBJ and will have different code associated with each occurrence of the symbol.

Clipper code

When compiled, each Clipper procedure or function in the .OBJ file has a separate unit know as a segment, which consists of a small assembly language header and a string of tokens. The header consists of pointers to the Clipper symbol table, tokenised code and a call to the CLIPPER.LIB procedure __PLANKTON. The tokenised code represents calls to functions within the Clipper library and parameters to those functions.

At application run-time, when the procedure or function is called, the __PLANKTON procedure processes these tokens sequentially and performs the appropriate library calls with the parameters held in the tokens. Each token is usually only one byte long, with parameters varying in length, eg a real number will take up 8 bytes and a character string will be stored as the length followed by the string. Tokens may also refer to symbols in the symbol table described above, rather than referring to absolute locations, so each reference to a variable will consist of a two byte symbol number. For example, the code

```
PROC T
A = B + C
```

would produce a symbol table containing the symbols 'A', 'B' and 'C'. The tokenised code would consist of (simplified):

```
Take symbol 2               (B)
Take symbol 3               (C)
Add them together
Store result in symbol 1    (A)
```

as shown in Figure 1.

This tokenised approach has a number of advantages over true compiled code, with only a negligible cost in performance. The code produced is very compact, for example taking only three bytes for a procedure call, as opposed to five for a direct call. It is also very self contained. All external references go via the symbol table, so operations such as incremental linking are made significantly easier. This approach also makes it possible to use the dynamic paging system described below for faster overlaid applications with lower memory requirements.

The size overhead of a simple Clipper compiled .EXE is actually made up of the run-time routines from the CLIPPER.LIB which are called by the processing of the tokens. The apparently large size of even a 'Hello World' type program is due to the potential for macro operations, which could execute just about any Clipper command from even a two line program.

Variables

Clipper 5.0 offers several different storage classes for program variables, depending on how they are declared and used in the program. Local and static variables are stored in a dedicated area of real memory, as described below. Private and public variables, known as MEMVAR variables, are created and destroyed dynamically while a

program is running, and are stored in Virtual Memory (VM) segments.

For performance reasons, these segments remain locked in real memory during most operations except memory-intensive operations and RUN commands. Each MEMVAR uses 20 bytes in a VM segment, so converting private and public variables to local and static variables can reduce memory requirements for some applications.

At run-time, each instance of a variable is allocated a value, which is represented internally as a data structure called a VALUE. The contents and format of a VALUE differ depending on the type of data it represents. Simple data, such as integers, are stored directly into the VALUE. Larger items, or data of variable length such as strings or arrays, have a 'reference' to the string or array stored in the VALUE, and the actual data is stored elsewhere.

Internally, Clipper is organised as a stack-based machine which uses an area of memory called the Eval Stack to contain temporary variables such as function parameters, intermediate results of expressions and local variables. The Eval Stack is simply a contiguous group of VALUEs that are accessed as a stack, in the same way as the processor stack is used by C programs.

For example, in a Clipper function call, parameters are pushed onto the Eval Stack before the function is executed. The function operates on the topmost items in the Eval Stack and produces a result. After the function completes, the parameter values are popped from the Eval Stack and replaced with the function result.

Each entry in the Eval Stack, ie each VALUE, occupies 14 bytes, and for complex data types such as character strings, arrays and code blocks there will be an additional memory requirement handled by the Virtual Memory Manager (VMM) where the actual value is stored.

The Eval Stack is allocated from the default data segment, defined as the start of the group DGROUP, when the program starts executing, so initialisation will fail if DGROUP is too full. This is not usually a problem with pure Clipper applications, but if a number of third party libraries are linked in to the application it may possibly fill up unless they have avoided storing data in DGROUP. The number of KB remaining in DGROUP for Clipper's use can be examined by executing the program, with the //INFO parameter, and the amount of conventional and expanded memory available will be displayed at the same time.

LOCAL variables are the simplest variables, and are allocated as locations within the Eval Stack to store their VALUEs. To manipulate a LOCAL variable, the system simply copies the variable's VALUE from one position in the Eval Stack to another. Local variables are visible only within the current procedure or function, and are created automatically each time the procedure in which they were declared begins executing. When that procedure terminates through a return, all its LOCALs are removed from the Eval Stack and any associated VMM memory freed up.

STATIC variables are similar to LOCAL variables, but have a duration of the lifetime of the application. Because of their permanence, they are allocated as fixed locations at one end of the Eval Stack, but are manipulated in the same way as LOCAL variables simply by copying their VALUEs. This means that every STATIC variable in the system also requires 14 bytes on the Eval Stack in DGROUP, which is another reason for C and assembler programmers to avoid storing data in DGROUP.

PRIVATE and PUBLIC variables are more complex than LOCAL or STATIC variables because, in addition to an associated VALUE, they also have a name which may be referred to during execution of the program via a macro or its equivalent. MEMVAR variables are allocated locations for their VALUEs in dedicated VM segments and these locations are stored with their names in the symbol table. When a MEMVAR is manipulated, the symbol table entry is used to point to the VALUE which can then be placed on the Eval Stack in the normal way.

FIELD variables differ from the other storage classes because they have no memory location at all, since their values are stored in a database record buffer. To manipulate a FIELD, the system generates a request to the file's database driver, which then creates an appropriate VALUE to be manipulated on the Eval Stack - see Figure 2.

Arrays

An array VALUE contains a reference to the array rather than an actual value, so when an array is assigned to a variable, the system simply overwrites the variable's VALUE with a new VALUE containing a reference to the array. The array itself is simply a group of VALUEs stored in virtual memory, where each element of the array is a VALUE. Any VALUE can contain another reference, so multi-dimensional arrays are created by having each element refer to another array rather than have an absolute value.

When values are assigned to array elements, the VALUE for that element is updated. When an array is assigned to another variable, only a copy of the VALUE referring to the array is made, and the array data itself is not duplicated.

Character values

A character string VALUE contains a reference to the character data, which is stored elsewhere in the VM. As with arrays, assigning a character value to a variable simply overwrites the variable's VALUE with a new VALUE containing a reference to the character data.

In a similar way to arrays, assigning a character value from one variable to another simply duplicates the VALUE (ie the reference to the data). The character data itself is not duplicated.

This reference-based memory management technique is the same for strings, arrays and code blocks. Clipper's garbage collector monitors references to objects, and when there are no longer any references to a particular piece of data, the space occupied is automatically reclaimed.

Macros

During program execution, when a macro is evaluated to the name of a variable or

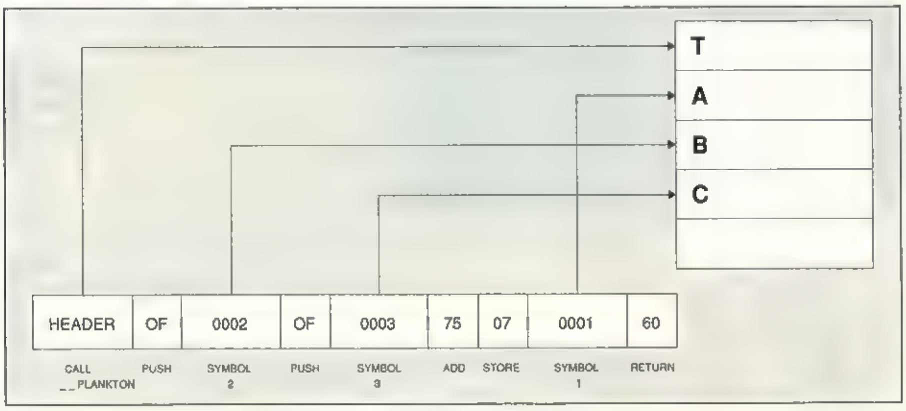


Figure 1 - A Simple Compiled Program

System Science

• Software • Specialists • Software • Specialist • Software • Specialists • Specialist • Software • Specialists •

Unix

SCO Unix Op. Sys **SCO** Unix Dev Sys.
SCO Open Desktop **SCO** ODT Dev Sys
SCO TCP/IP & NFS **FTP** PC/TCP
Interactive Unix Op Sy / Architect Series
Informix 4GL & SQL **RM** Cobol-85
LPI Compilers **C Scape** Screen Lib
LPI C++ (native) 386 Unix & Sun- **NEW**
... many more for 386 Unix and Sun

Mathematics

Derive - The Mathematical Assistant
Mathematica 2 Windows (now in), 386. MAC
Mathematica for Workstations (Sun, HP..)
MathCad for Windows (new) GrafTool
What's Best! **Lindo**
SPSS/PC, Statgraphics Chiwriter Sci WP

Windows Development

(see Microsoft , Borland, Zortech, Watcom)

CASE:W Corp.	£645.00
Smalltalk/V Windows	£295.00
Btrieve for Windows	£345.00
Object Trieve	£call
Commonview	£395.00
Win++ by Blaise	£150.00
Windows Control Pallette - Blaise	£95.00

CrossDevelopment

2500AD Cross Assemblers	£135.00
2500AD Simulator/Disassemblers	£115.00
Cross Compilers - Introl, Manx, Hitech	

Pascal (see Borland)

TopSpeed Pascal SE	£115.00
Object Professional	£110.00
Btree Filer MU	£120.00
Asynch Professional	£89.00
Blaise Turbo Vision Toolkit	£89.00
Blaise Turbo Asynch Plus	£115.00
PCX Programmer's Toolkit	£155.00

Power Basic	£89.00
many Basic libraries for comms, graphics	
PC Logo	£50.00
LMI UR/Forth	£295.00
Smalltalk V/DOS	£69.00
muLISP 90	£225.00
RM Cobol-85 (new Dev Pack)	£call.
MS DOS 5.0,	£call.
QEMM 386	£65.00

Microsoft

MS C/C++ 7.0 **NEW!...NEW!**	£225.00
Quick C for Win **NEW!**	£99.00
MS Fortran 5.1	£210.00
MS Windows SDK	£210.00
MS Basic Compiler 7.1	£225.00
MS Cobol 4.5 (DOS & OS/2)	£460.00
MS Macro Assembler 6.0	£79.00
MS Visual Basic	£99.00
MS Quick C 2.5	£60.00
MS Quick C & Assembler	£99.00
MS Quick Basic	£60.00

C/C++ 7.0 competive upgrade £99.00
call for details

Authorised Languages Dealer

Borland

Borland C++ & AFW **new 3.0**	£295.00
Borland C++ (Dos / Win) **3.0**	£195.00
Turbo C++ Windows **NEW!**	£85.00
Turbo Pascal Windows	£99.00
Turbo C++ & Turbo Vision **NEW!**	£75.00
Turbo Pascal 6	£75.00
Turbo Pascal 6 Prof	£135.00
Turbo Debugger and Tools	£89.00

Authorised Languages Dealer

Nu-Mega Tech.

Bounds Checker (386)	£169.00
Soft-Ice/W (Windows)	£259.00
Soft-Ice (for 386 DOS)	£259.00
Magic CV 3.0	£135.00

Intersolv *(Authorised)*

PVCS for Dos	£345.00
PVCS Config Build (PolyMake)	£145.00
PVCS Networks, OS/2 & Sun	£call
PVCS Professional Editor (SPE)	£155.00

MKS

MKS Toolkit new ver 3.2	£155.00
MKS RCS new ver 5.1	£145.00
MKS Lex & Yacc new ver 2.6	£145.00
MKS multi-user and OS/2	£call

Tools and Editors

Brief 3.1	£195.00
Kedit (Xedit for PC)	£99.00
.RTPatch for DOS, Win & OS/2	£195.00
.RTLink Plus	£295.00
Plink 86 Plus	£275.00
Personal Rexx new ver 3	£110.00
Ghost (auto software testing)	£135.00
EasyCase Plus	£345.00
C Programmers Toolbox	£175.00
CDOC	£145.00

... and many more tools and utilities

C++ Compilers

also Microsoft & Borland

Zortech C++ Dev Ed (incl OS/2)	£345.00
Zortech C++ Sci & Eng	£525.00
JPI TopSpeed C++PExtend	£175.00
JPI TopSpeed C++ SE	£115.00
JPI TopSpeed C++ PE	£175.00
JPI TopSpeed C++ Pro Dev.	£229.00
Watcom C 8.5	£275.00
Watcom C386 8.5 with ADS Supp	£495.00
Aztec C 86 (with ROM support)	£265.00
TopSpeed C PE	£175.00

Borland C++ & AFW **only £189.00**
for owners of MS-C, Quick C, QC Win,
Zortech C++ & JPI ...call for details

C Datafile

CodeBase 4.5 for C, C++, Win- new	£235.00
Btrieve - DOS or Windows ver	£345.00
Ctree Plus from Faircom	£345.00
Faicom Professional Toolbox	£745.00

C Communications

Essential Comms	£175.00
Greenleaf CommLib	£195.00
C Asynch Manager (Blaise)	£115.00

C Screens

CScape with Look & Feel	£365.00
Vermont Views 3	£335.00
Panel Plus II	£275.00
Zinc for C++ (Win &Dos)	£225.00

C Graphics

Essential Graphics & Chart	£195.00
PCX Programmers Toolkit	£155.00
Halo	£195.00
Metawindows	£195.00

... many more libraries

Pharlap **386** DOS Ext SDK	£325.00
Pharlap **286** DOS Ext SDK (new)	£325.00

Fortran Compilers

Lahey F77L	£375.00
Lahey F77L-EM/32 with OS386	£875.00
Watcom F77 386 8.5 with ADS supp	£495.00
Salford FTN 77-386	£765.00
PC/Interacter (screens)	£325.00
Ingraf Graphics source	£195.00

... many Fortran Libraries

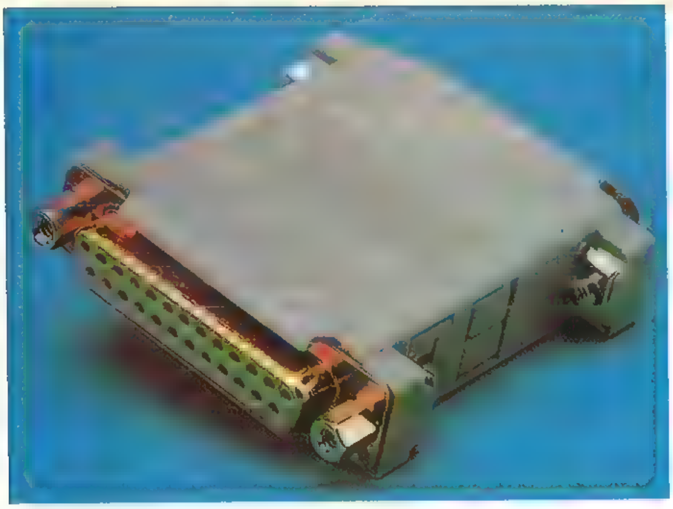
- **Prices are exclusive of VAT.**
- **Prices include delivery to GB.**
- **Prices are subject to change.**
- **VISA, Access and Mastercard welcome with telephone orders.**

3-5 Cynthia St
London N1-9JF
Fax: 071 837 6411

071 833 1022

• Software • Specialists • Software • Specialist • Software • Specialists •

CIRCLE NO. 855



How many users of your software paid for it???

SOFTLoK International Limited was established in 1987 with the introduction of our SOFTLoK and SOFTLoK PLUS devices to combat the ever increasing problem of software piracy. Our SOFTLoK range of software protection devices are used by hundreds of software developers from small consultancies to large multinationals. To cope with the ever changing needs of our customers we have developed SOFTLoK II which combines the programmable features of SOFTLoK PLUS with a low unit cost similar to our original SOFTLoK product.

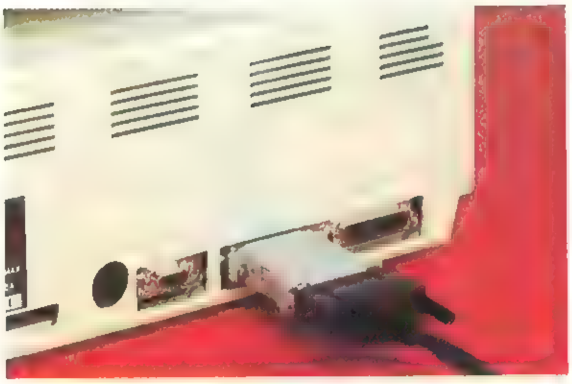
SOFTLoK II™

The Next Generation

SOFTLoK II units are programmable devices containing read/write memory protected by a password. Both the memory and the password can be changed at any time using our routines in your application software. Easy to use menu-driven software is provided to allow small or large batches of SOFTLoK II units to be programmed with their initial data & passwords ready to be sent out with the protected software product.

Price: 1-19 £16.90, 20-49 £15.40,
50+ £13.70

Evaluation kit £20 (SOFTLoK II, manual & software)
All prices excl. VAT and delivery



As SOFTLoK II units plug into the parallel printer port they can be installed or removed in seconds.

□ For IBM PC, PS/2 and compatibles
□ Uses parallel printer port
□ Totally transparent to printer
□ Secure data & password can be changed from your application software
□ Cascadeable
□ 240 bytes of secure read/write memory
□ 8 byte (64 bit) password
□ No programming adaptors required
□ Easy to use SOFTLoK II setup software
□ Routines ready to link with various compilers
□ Easy to follow manual

I-MEX House, 40 Princess Street,
Manchester, M1 6DE, England,
Tel: 061 228 7379 **Fax:** 061 236 6890

CIRCLE NO. 856

procedure, the symbol table is searched to find the requested name. Once the name is found, Clipper follows the pointer in the symbol table to the VALUE where all the general information about the symbol is actually stored.

The VALUE will indicate whether the procedure or variable being referenced has been defined, and Clipper checks this before continuing any further. If it is undefined and is not a variable being created, Clipper immediately returns an appropriate error - 'missing external' for procedures or functions, and 'variable does not exist' for variables. If the procedure or function has been defined correctly, then the VALUE will contain a pointer to the program code to execute for that procedure, and control can be transferred to the procedure.

The remaining case of creating a new variable is handled by adding a new entry to the end of the symbol table. This new entry will have the name of the variable filled in, along with a pointer to a VALUE for the symbol, and will be used from then on to refer to the variable.

Both Summer '87 and Clipper 5.0 provide other mechanisms to avoid the creation of these dynamically named variables in the majority of circumstances, such as using an array of elements to store the values, or using code blocks in 5.0. These alternative mechanisms should be used wherever possible, if only because macro operations are inherently very slow, as each name in the symbol table has to be checked until a match is found before execution can continue.

If the use of a macro cannot be avoided, but the name to be created will be one of a known set, then these names should be mentioned explicitly somewhere in one of the programs. The code does not ever have to be executed, but just using the names causes them to be added to the symbol table at compile time, thus avoiding the above situation.

Code blocks

Code blocks are represented internally as strings of tokenised code, in the same way as normal procedures and functions. When a code block is assigned to a variable at run-time, a pointer to the tokens making up the code block is stored in the variable, along with information pertaining to the currently active procedure.

Because the code block consists of normal tokens, it will include references to the symbol table, so the equivalent symbol table must be available when the code block is actually evaluated. This is one of the reasons why it will prove difficult (but not impossible) to save code blocks in a database from one application and restore and evaluate them at a later time in the same or another application.

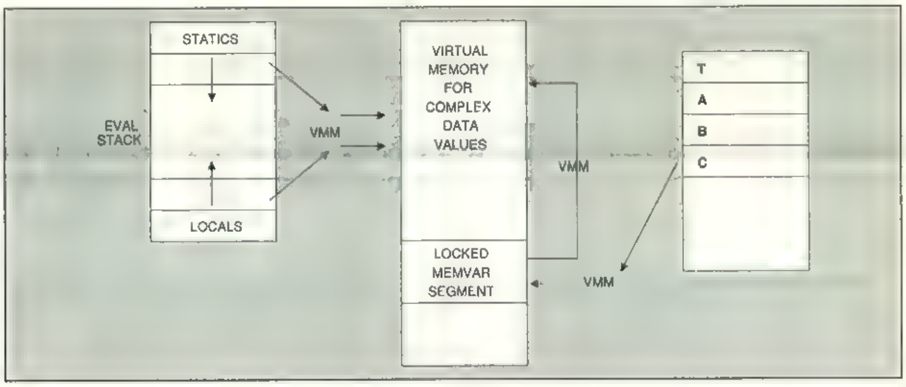


Figure 2 - Storage of Data Variables

Dynamic paging

When linked with BLINKER or .RTLink, Clipper 5.0 performs its own form of dynamic overlaying of compiled Clipper code, which results in extremely fast, memory efficient execution of the code.

During linking, all Clipper modules are broken down into pages of 1 KB. These pages are stored either in the executable file or in separate overlay files. The manipulation of overlays in these 1 KB pages removes the effect the size of compiled functions or modules has on the memory required to load the overlay. Large modules are broken into multiple pages, and small functions are grouped together in a single page.

At execution time, Clipper 5.0's dynamic overlay manager loads pages based on information embedded in the .EXE by the linker. The dynamic pages are loaded into VM (Virtual Memory) segments, allowing the VMM to manage the overlay pages on a competitive basis with other uses of memory such as the application data.

The paging architecture allows the system to discard low-use sections of code even if the code is still active, and reload it only when control returns to that piece of code. Code pages which are being heavily used are maintained in memory by the VMM's LRU swapping policy.

When possible, the VMM will place dynamic overlay pages in expanded memory, reducing overlay reads. Overlay pages are never written to the VMM disk swap file, however. If a VM segment containing an overlay page is to be removed from memory altogether, it is simply discarded. If it is needed subsequently, it is reread from the overlay file. In addition to virtual memory, the dynamic overlay manager uses a dedicated area of real memory to cache the most active dynamic overlay pages.

This page mechanism is made possible by the nature of the Clipper code. As explained before, it is not actually code but a series of tokens which are processed at run-time. This means that the __PLANKTON procedure from CLIPPER.LIB can detect when it has reached the end of a page and request the next one to be loaded.

All Clipper code is therefore overlayable, so there are no restrictions on which Clipper .OBJs can be placed in the overlay area. It should be noted that linkers which use the dynamic paging mechanism of Clipper 5.0 automatically overlay *all* Clipper code unless directed otherwise.

Summary

We have discussed how a Clipper program is converted into an executable program by creating procedures and functions consisting of tokens referring to a symbol table, which in turn refers to the actual code or data associated with the particular symbol.

We have also discussed the way Clipper manages these types of code and data in memory at application run-time, which should give the developer a better understanding of the underlying operations during execution of a Clipper program. This in turn should enable the developer to write more efficient programs, both in terms of execution speed and memory requirements.

EXE

Jud Cole has been programming on microcomputers since 1979 in many languages including Assembler, C, Pascal, PL/I, dBASE and Clipper. During 1989 and 1990 he developed Blinker, the first dynamic overlay linker, and founded Blink Inc to market it. Blinker is distributed in the UK by QBS Software (081 994 4842).

The disk cache

The UNIX file system is good at insulating programs from track-to-track seek times, but there is a price to pay for such insulation. Peter Collinson explains.

The designers of UNIX were among the first to realise the importance of the object that we know as a 'file'. They provided a system designed to allow you to create files quickly and easily. It's hard to look back and appreciate what a great change that was.

A key part of the design was the 'buffer cache'. Any data read from the disk is retained and passed to as many processes that want it. If a process writes to the disk then the data is not moved physically onto the media but it is kept around in memory. The data finds its way onto the disk when the space in memory is needed.

Cache problems

In general, the file cache speeds access to the file system. The down side is that the kernel holds 'state'. At any time, the file system structure and contents are stored on the disk and also in the memory of the machine. If the memory should go away - the machine is taken down - then the file system on the disk may be incomplete.

Taking the machine down is a controlled activity. We provide a system call, `sync`, that ensures that the cache is written to disk. Users often see this as a utility, also called `sync`, typed just before halting the machine. Traditionally, the command is typed twice. The first one flushes the cache, the second one gives you something to do while you are waiting for things to hit the disk surface.

In addition, the system runs a background process called `update`. This invokes the `sync` system call every 30 seconds. It improves the chances of the file system being OK if there is a sudden unexpected system crash like a power fail or (heaven forbid) a system panic. If the system has been quiet just before the crash then the disk will be consistent and all will be well.

However, the unexpected usually happens when the machine is at its busiest. Even typing `sync` twice before halting the machine is no security if there is massive disk activity because there are many processes running. A crash or a halt on a busy system can mean that the file system structure on disk needs repair.

There are many easy cases where just minor repairs are needed. It may be that a file has just been deleted but its contents are not on the list of free blocks. There are worse cases. The system may have been changing the structure, perhaps adding or deleting a directory. The structure is no longer consistent. Further use may compound the problem, resulting in the total loss of precious files.

Inodes

To get some further idea of the problems, let's look at the creation of a single file. We are executing some simple system call like:

```
fd = creat("fred", 0666);
```

On many systems, this is no longer a system call. For the moment, just assume that it is. The call creates a file `fred` in the current directory. It will be set to file mode `-rw-rw-rw-` depending in the value of the user's `umask`. The file will have zero length. The file will be truncated to zero length if it exists when the call is made.

The system call will return with a file descriptor, a small positive integer. Actually, this is the index into a table of 'file table' entries. The file table allows several processes to point at a file and each to have a separate position pointer. The file table also points to an in-memory copy of the index node or *inode* for the file. Each file on the file system is referenced by an inode.

When the system wants to use a file, it brings the relevant inode into memory using its number to index a table stored on the disk. Among other things, the inode contains the file information returned by the `stat` system call. For example: the owner information, the various access times and the mode of the file are all stored in the inode for the file. It also contains the block addresses on the disk where the file contents will be found. It doesn't contain them all, because otherwise the inode would be variable length.

The inode doesn't hold the name of the file. It is the job of a directory, in fact the whole directory structure, to tie a pathname string to the inode number. This allows links in the file system, several directory entries pointing at the same inode entry.

The inode does contain the size of the file. This means that the size is not determined by its contents or by the blocks that it occupies. The size of the file is extended by the `write` system call and the `lseek` system call. It can be altered up or down by the `truncate` system call. When reading, the system compares the read pointer in the file table with the size of the file in the inode. If the file pointer is larger than the size, the system will return an end-of-file indicator.

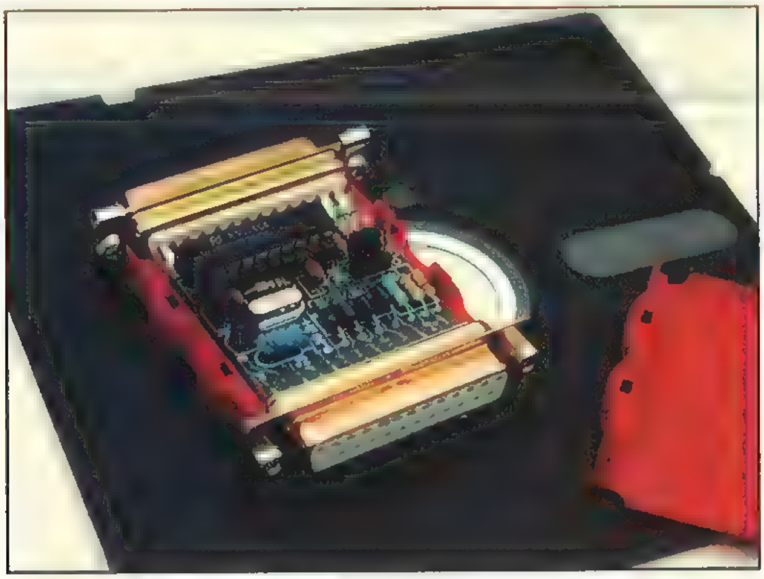
You get the picture, I hope. The job of the `creat` call is to return a handle to a file table that will have a position pointer for the file (set at zero) and also a pointer to the inode for the file. The inode pointer references the file on the disk. It contains system information and the addresses of the blocks of data on the disk. Of course, for our `creat` call, there will be no data blocks.

The creat call

Having seen a little how the file system works, we can now look at what happens when the `creat` system call is executed. The first argument to the call is the name of the file to be created. I have selected a simple filename `fred`. The file is to be created in the current working directory. This is an in-built assumption, since the filename contains no path information.

The kernel stores a pointer to the inode for the current working directory as part of the information that it retains for the process. The first task of the `creat` call is to scan the current directory looking for a file of the required name. The blocks that comprise the current directory are read into the kernel. Each directory entry is scanned, searching for a file name that matches the name we are looking for.

Announcing DeadLock III



Don't think Dongle, Think DeadLock

- Unbreakable
- Price unbeatable
- Completely new system
- Approved as the highest security available

Technical

- Sits on parallel port
- Completely non-intrusive and transparent
- Programmable
- The hardware component is based on a custom-designed IC
- There's full DeadLock support for today's most popular languages, compilers and development environments
- DeadLock contains all the logic necessary for password control and to perform calculations
- DeadLock comes with a special PROTECT utility that allows you to build an 'envelope' of protection around your product's COM or EXE files
- Each DeadLock key can be programmed by you, to transform it into a protection device that's unique to your product
- Easy to use manual
- May be used for joint ventures
- DeadLock comes with a variety of friendly, menu driven utilities
- Different coloured cases available to suit your preference

Distributors required worldwide. Please send CV to address below.

Please address all enquiries to:

BL Computer Security Limited
101 Hendon Lane, Finchley, London N3 3SH
DeadLock
PO Box 2543, London N3 3UA, United Kingdom
Fax: 081 346 2672 Tel: 081 343 0734 Northern Ireland Tel: 0232 682047

CIRCLE NO. 857

We have two cases, either we find the file - or we don't. If we find the file, then the scan will succeed. The directory contains the inode number that we need to reference the file. We pull the inode contents into memory. We need to free any blocks associated with the file, and update the disk copy of the inode to show this. We also will update the access time on the current directory - this will involve writing the in-memory value out to disk. We are done and return this inode, pointed to by a file pointer.

If no file is found that matches the name, a new file needs to be created. A new file needs a new inode. Usually the kernel keeps lists of free inodes for each mounted file system so this is reasonably cheap.

The free inode is found and an in-memory copy will be established. It will be loaded with the various bits of relevant information, the owner will be derived from the process image, the access mode from the second parameter to the `creat` call and so on. We have an inode and we need to add the new name to the directory.

A free slot in the directory is usually found earlier when looking for the name. The relevant disk block is brought into memory and updated with the new file name and its new inode number. It will be written out. We have just accessed and modified the current directory, so we need to update the disk copy of its inode. We also need to update the disk copy of the inode for the new file.

As you can see, in either of these cases, the apparently trivial system call involves a number of alterations to the file system. We haven't actually written any data yet. There are several points in the creation of a file where the file system is unsafe. By the way, please don't take my sequencing above as the gospel truth - I am going through the operations from a logical viewpoint.

File system repair

If the system crashes, what then? We must verify the disk structure to ensure that it is coherent before we can run a live system. Luckily, we have a program that will do this for us, it's `fsck`.

The operation of `fsck` is reasonably simple, although it obviously varies depending on the file system. The program consists of several phases that make checks on different aspects of the file system.

The first phase checks the inode information. It verifies that the inode types are correct. It ensures that the inode size is not less than the number of blocks allocated to the file and checks that the disk addresses that are held in the inode make sense.

During this, it builds an allocated block map of the file system. This is used to ensure that no disk block is pointed to by more than one file. These are called 'Dup' (for duplicated) blocks. Two files pointing at the same disk block is bad news, as one file of the pair will contain rubbish. Problems in this phase will result in the clearing of an inode losing any data that it might contain. The user is asked to confirm that the action should be taken.

Some systems adopt the policy of not running fsck on file systems known to be safe

The second phase checks pathnames and directory structure. It will remove any entries that point at inodes that were cleared in the previous pass. It will find directory entries that point to unallocated inodes and clear them away. It will validate that the '.' and '..' entries in the directory contain sensible values. It will check that the directory contents make sense and contains what appears to be valid data.

The third phase will check connectivity. This section picks up errors that may have been found in phase 2. It will find unreferenced directories and can optionally reconnect them in the `lost+found` directory.

The fourth phase checks reference counts. This verifies the link counts that were seen in pass 2 and pass 3. It's here that files that are unreferenced by directories are picked up and either reconnected in `lost+found` or discarded. It will also fix up any bad link counts on directories.

Finally, the last phase will scan the block maps and rebuild the free block lists. This cleans any of the debris left by files that have been cleared.

At the end of the run, you should have a repaired and usable file system. The program has a 'preen' mode of working that will fix 'easy' errors. In general, these errors are those that do not affect file contents: unreferenced inodes, link counts too large, missing blocks in the free list, blocks in the free list also in files and wrong counts in the file system super-block. It is usual to use preen mode at system start-up.

If preen mode fails, then you run `fsck` hands-on to fix the file system. It's usually safe to answer 'yes' to any question that is asked. This will get the file system back to a usable state, even though it might lose files. Beware that questions will cascade. For example, answering 'yes' to clearing an inode may result in having to clear a directory entry later in the run. I never clear an inode that points at a directory. It's better to try to reconnect it in `lost+found`. Later, when the system is live, you can get into `lost+found` and delete or save things as needed. If you don't reconnect the directory, you may find yourself hand clearing many files and directories.

If I get errors, I will always rerun `fsck` until the partition checks out as being clean. This seems a good rule to follow.

Problems with fsck

The main problem with `fsck` is that it is slow. Human impatience means that it can be omitted from the regular reboot sequence to 'save' time. Things in the file system can get so corrupted that the program cannot fix it. It's good system administration to run `fsck` whenever the system is rebooted.

Some systems adopt the policy of not running `fsck` on file systems that are known to be safe. Remember that a UNIX tree structured file system is generally made by 'mounting' several partitions to form a coherent tree. When the system is taken down, any file system that is unmounted cleanly is marked as safe and is not checked by `fsck` later. It's not possible to unmount a file system if it is busy, and busy is defined by very strict rules. No process may have any file or directory open for any activity on the file system. Because no-one is talking to the file system, it is unmounted. It follows that the disk must have a coherent structure and `fsck` is not needed.

Another problem is one of synchronisation. Each disk on UNIX is provided with two kernel interfaces. The block interface is used by the mount system call to access the file system on the disk. The disk cache sits below this. The character interface allows access to the 'raw' disk and means that the system will use DMA to inject disk data directly into the address space of any process using it.

Developing software on a LAN is fast, efficient, and dangerous.

We make it safe, controllable and auditable.

Experienced managers are rightly concerned about the integrity of LAN development environments. It's too easy to overwrite or corrupt files.

To safely develop on a LAN you need control over file changes, the ability to recover any module revision or system version, audit trails, and security that doesn't hamper productivity.

The solution: our LAN development environment, three tools that operate independently or snap together synergistically.

The primary element is PVCS Version Manager which provides version control and change management features, files and function security, automatic audit trail and detailed project reports.

The best way to access PVCS Version Manager is via our PVCS Professional Editor, an extensible control centre that simplifies access to any tool and provides industry strength editing. PVCS Professional Editor invokes PVCS Version Manager automatically. Just open modules to edit and the PVCS Version Manager window pops open.

The final element in our LAN development environment is PVCS Configuration Builder - a sophisticated tool for automatically performing correct builds. A single command recompiles all modules that have been changed, re-links as necessary, and produces a correct, conforming program.

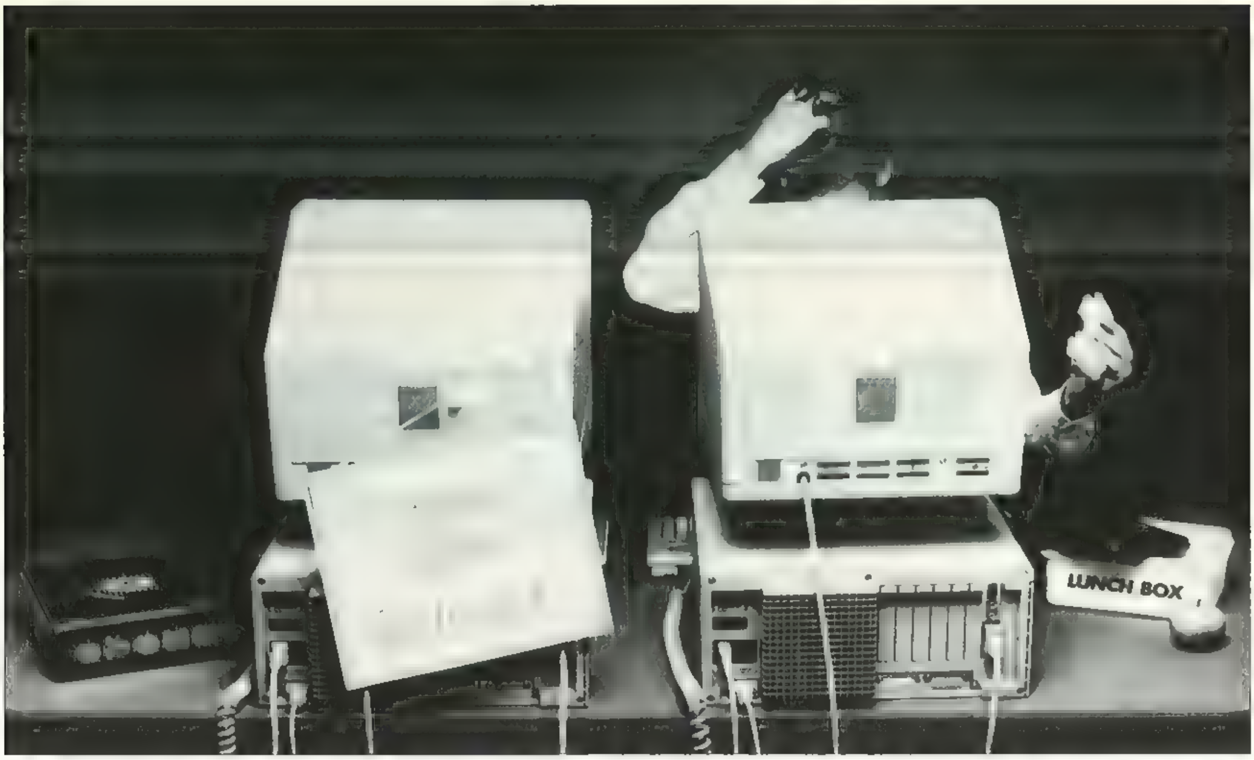
Don't Let These Special Offers Pass You By.

Special Offer Prices are valid until April 30th 1992

	DOS or OS2	DOS and OS2
PVCS Version Manager	£315	£416
PVCS Configuration Builder	£136	£180
PVCS Professional Editor	£160	£208
Mainframe Gateway	£112	--

Software Express · Portmill House, Portmill Lane, Hitchin, Herts SG5 1DJ. Tel: 0462 422525

CIRCLE NO. 858



Sycero from System C. The best thing on the programmer's menu

Sycero is a powerful 4GL and applications generator for dBASE, Clipper or C programmers.

Sycero is powerful enough to create even the most sophisticated database applications. In fact, it gives you the best of both worlds: the speed and ease of use of a 4GL with the flexibility of a 3GL.

Sycero's complete development environment consists of a data dictionary, screen and report painters, a powerful 4th generation language based on the dBASE/Clipper standard. It also includes automated program types such as menus and file maintenance, reducing the amount of programming needed for common tasks. Networking code can be automatically generated and high quality documentation is produced as standard.

The multi-file report generator lets you build the simplest of lists or the most complex invoice, and automates sub-totals, pagination, report loops, user selection of ranges and sort sequences and wildcard matching.

So, if you want to stay ahead in PC database development and **still** have time for lunch, call System C now on 0622 691616, or return the coupon below.

To: System C Ltd 60-61 High St Maidstone
Kent ME14 1SR
Please send me further details about your Clipper generator, Sycero dB □
Please send me further details about your C generator, Sycero C □
Name ....
Company ....
Address ....
....
.................................Postcode ....
Tel ....Fax ....

EXE 4/92

System

System C Ltd 60-61 High St Maidstone Kent ME14 1SR
Tel 0622 691616 Fax 0622 691241

CIRCLE NO. 859

dBASE is a trademark of Ashton-Tate. Clipper is a trademark of Nantucket.

To make `fsck` run faster, it normally uses the character system interface. This is faster because we avoid copying data in from the disk and out to the user process. However, if the disk is mounted when we run `fsck` then we can have some potential problems.

The main problem is that we are looking at the file system with two different pieces of software, `fsck` and the kernel. Either can alter the file system structure without knowledge of the other. This isn't good news.

The disk cache will contain the 'correct' view of the file system. It's necessary to `sync` the disk to ensure that the cache contents are written out. Even then, `fsck` may alter the file system behind the back of the kernel. So, it's good practice never to run `fsck` on a mounted partition.

On your system, you may notice an automatic reboot just after the root partition is repaired. This is because it must be mounted when the system is booted. If there are problems with the file system integrity, then they are fixed and the system rebooted. None of the other file systems should be mounted when `fsck` is run.

I once came across a site that was running `fsck` to 'check' the disks on a nightly basis from `cron`. I was unable to persuade them that this was not how fsck was intended to be used and worse, that what they were doing was more likely to cause damage than prevent it.

No, they had had a bad experience with the disks being in a very bad way on one occasion. This hadn't happened since they started the nightly checks so the checks were the right thing to do. They were also under the impression that the UNIX file system suffered from rot, it got more corrupt as time went on. The only thing that was saving them from disaster was that they all went home at 5.30pm and no-one ever used the machine after hours.

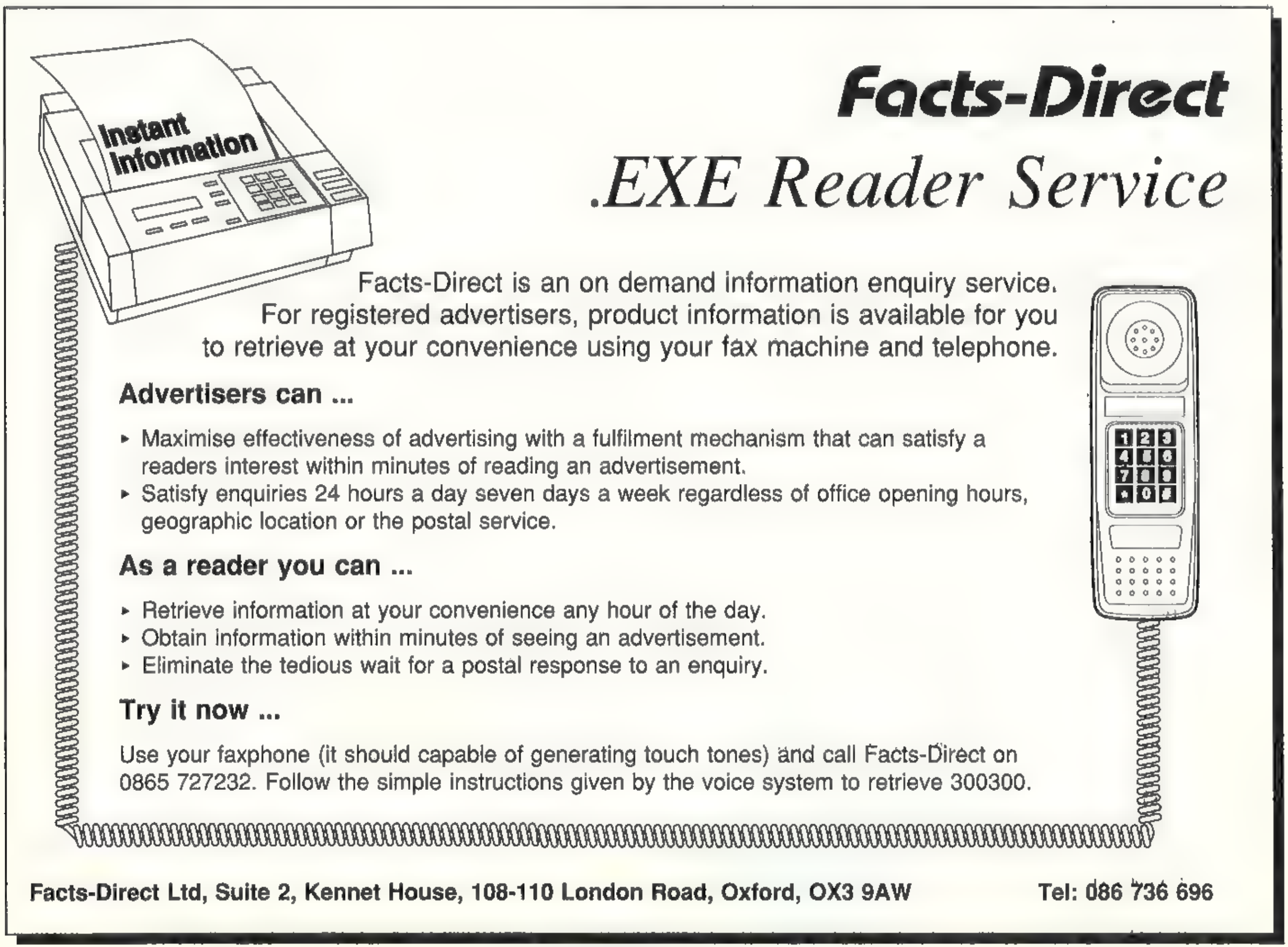
Finally

I don't want to leave you with the impression that the price you pay for checking the disks with `fsck` is greater than the benefit of having a disk cache. The purpose of this article was to try to fill in some of the confusion that exists in this general area.

The cache is a huge speed win in most operational circumstances. The benefits of decoupling the processes from running in synch with the disk are immense. It has stopped the need for RAM disks on UNIX systems for a long time.

These days people have found that RAM disks for the `/tmp` are useful because they speed up file creation. Most systems now insist that the inode for a new file is written to disk before they attempt to modify the directory to point at the new inode. This means that there is a disk write synchronous with the calling process whenever a file is created. This can slow things down when a program is creating many files. Creating this in RAM speeds things up considerably.

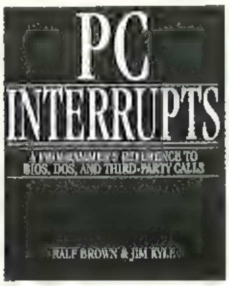
Peter Collinson is a freelance consultant specialising in UNIX. He can be reached electronically as `pc@hillside.co.uk` *(although your mailer might be happier to put the address the other way round) or by phone on 0227 761824.*



Books

This month, a popular bulletin board file achieves a higher form.

You've read the file...


PC
INTERRUPTS
A PROGRAMMER'S REFERENCE TO BIOS, DOS, AND THIRD-PARTY CALLS
RALF BROWN & JIM KYLE

In the exe conference on CIX, placed there by a stalwart code-named `hbroomhall`, and on various other programmer-oriented bulletin boards conferences such as CompuServe's IBM Programming Forum, there is a collection of files called the *MS-DOS interrupt list.* Consisting of, in the current version, about 600 KB of compressed material, these files detail hundreds of MS-DOS interrupts.

You may say: 'But there *aren't* hundreds of MS-DOS interrupts'. *Au contraire.* As well as the hardware interrupts, the BIOS calls and the standard MS-DOS functions, there are all those small, but immensely useful bits of information which tend to get left out of *Eddy Expert's Programmers' Guide to MS-DOS* (£43.50 with disk). The INT 33H mouse driver calls, for instance.

An American named Ralf Brown noticed that all these bits of information could beneficially be corralled, so he started this list, which has now been published as *PC Interrupts.* His philosophy seems to be: get everything in. So we have not only standard published DOS and BIOS interrupt calls, plus mouse, Windows and so on, but also 'undocumented' DOS (Brown and his co-author Jim Kyle also worked on last year's hit book *Undocumented DOS*) and loads of proprietary stuff from the VESA SuperVGA BIOS to STSC's APL *PLUS/PC, from Novell Netware to the SoundBlaster SBFM Driver and from the Nina virus to the Solano virus.

The information given per function is necessarily minimal - as it stands the book is over an inch and a half thick. Each function is documented with a name, a one or two line description of its purpose, input and output registers, conflicts and descriptions of any data structures that are required. But unless you are very ambitious, you will find that the information here is insufficient for your needs.

PC Interrupts is an excellent reference book, and deserves shelf-space wherever DOS application programmers find themselves obliged to fool around with system-level detail to get their programs to work - ie nearly everywhere.

Title: PC Interrupts	*Authors: Ralf Brown and Jim Kyle*
Pages: (about) 500	*Price: £27.95*
Publisher: Addison Wesley	*ISBN: 0-201-57797-6*

Books Received This Month

Unix Pocket Book	Steve Heath	Butterworth-Heinemann	£12.95	ISBN: 0-7506-0391-7	406
Pascal and Beyond	Steve Fisher & Stuart Reyes	John Wiley	£17.95	ISBN: 0-471-52996-6	496
Microsoft Win32 API, Vol 1	Microsoft	Microsoft Press	£34.95	ISBN: 1-55615-497-6	677
Microsoft Win32 API, Vol 2	Microsoft	Microsoft Press	£34.95	ISBN: 1-55615-498-4	717

The Clinic

Mike Banahan introduces a new, regular C++ clinic, this month concentrating on member initialisation.

This is the first of a regular series of C++ clinics - readers are encouraged to write in with their questions. In general we'll be sticking to questions of the C++ language itself. Asking how to get up the floating point register window in a particular vendor's debugger is *not* the kind of question we'll be answering; on the other hand, it's entirely fair to ask whether you should be using

```
delete [n] p;
```

or

```
delete [] p;
```

no matter whose compiler you are using.

Be warned though - the answers will be the best that we can come up with. Guarantees of correctness are a different matter altogether. If we're wrong, then tell us!

Clinic #1

It's fascinating to see how many people insist on programming in the style of older languages, rather than using the bits of C++ that are there for a good reason. Whatever your opinion of the overall elegance of the language, it is very rare to find that any of it is there gratuitously. Most of it has been thoroughly thought through - even if the reasons for some it aren't all that obvious. Member initialisation is one of the poorly-understood parts of the language, and there's something to watch out for here. Avoiding it for too long is unwise. Let's start by looking at what it is.

Any member of a class (or struct for that matter) can be initialised in a constructor, rather than by being assigned to. In many cases this is simply an option and it doesn't matter which you choose:

```
class example1{
  int myvariable;
public:
  // Constructor
  example1(int initval){
    // assign to myvariable
    myvariable = initval;}
};

// Or alternatively
class example2{
  int myvariable;
public:
  // Constructor - uses
  // initialisation
  example2(int initval) :
      myvariable(initval){}
};
```

Sometimes you absolutely must initialise certain members. The 'most' obvious case doesn't actually look like member initialisation at all; it's when you need to pass arguments to a class's base(s) when inheritance has been used.

```
class parent{
public:
  parent(int a, int b);
};

class child: public parent{
public:
  // Pass parameters to the
  // constructor for "parent"
  child(int x, int y) :
            parent(x,y){}
};
```

Just to complicate matters, if the derived class only has one base class, it can omit the name of the base; although valid, this is poor style.

```
class child: public parent{
public:
  // Pass parameters to the
  // constructor for "parent"
  // using unnamed member style
  child(int x, int y):(x,y){}
};
```

Apart from inherited base(s), which other members of a class must be initialised from within the constructor? Any non-static `const` or reference that the class holds.

```
class hasconstandref{
  const int         ci;
  int               &ir;
public:
  hasconstandref(int cinit,
                 int &rinit) :
    ci(cinit), ir(rinint){} };
```

Incidentally, the constructor doesn't have to be inline - the examples have only chosen that for compactness. Here's the non-inline version of the one above.

```
class hasconstandref{
  const int         ci;
  int               &ir;
public:
  hasconstandref(int cinit,
                 int &rinit);
};

hasconstandref::
  hasconstandref(int cinit,
                 int &rinit):
  // Initialisers
  ci(cinit),
  ir(rinint){
}
```

Opinions differ as to what makes an acceptable style for the layout of constructors with that kind of initialiser list *(and our layout has been distorted by .EXE's narrow columns - Ed).*

Of course, the issue is easy if you *have* to use initialiser syntax. What should you do when you've got the choice? Oddly enough, it may really matter (although not in the cases that we've shown). Think of the case where a class member is itself an object and is blessed with non-trivial constructors and assignment operators.

In that case, it may matter a lot. If you set the object's value by using assignment within the body of the outer class's constructor, then a lot more work needs to be done. First, the inner object has to have its own constructor called (perhaps lots of work). Then, its assignment operator is invoked to set its value (lots more work). If you could do all the work in its constructor, you could easily obtain a measurably more efficient operation.

As a rule of thumb, whenever you can use initialiser syntax you should, even if assignment is possible. Initialiser syntax is there to provide initialisation; choosing to use assignment should be deliberate either A) because assignment is known in this case to be more efficient, or B) because the semantics of assignment and initialisation are different and assignment is the one you want.

The last case raises some worries about the quality of your design, but may just be unavoidable. Quality of design is a very deep question indeed...

Mike Banahan is Chairman of ECUG. Subscription to ECUG is £50 per annum. For more information about its activities, contact Rebecca Thomas on 071 253 5121, or write to ECUG, c/o City House, 190 City Road, London EC1V 2QH.

To advertise in this section, please call Marc Warren on 081-994 6477. Fax 081-994 1533

Give *Real-Time* Projects a Head Start
with the AMX™ low-cost high performance Multitasking Kernel

Features

Fast, reliable operation
Compact and ROMable
PC Peripheral support
DOS file access
C and Assembler supported
Preemptive task scheduler
Time slicing available
Configuration Builder
Intertask messages
Message exchanges
Dynamic operations:
- create/delete tasks, timers
- adjust task priorities
- memory allocation
Buffer Manager
Semaphore Manager
Breakpoint Manager
Event Manager
List Manager
Sample and Demo Programs
InSight™ Debug Tool for use with most popular debuggers

Targets

AMX 86 for real mode 80x86

AMX 386 for protected 80386

AMX 68000 for 680x0

AMX 80 for Z80, 64180 and 8085

No royalties

Source code included

For more information and a *free* demo disk contact:

ScotWare Ltd
45 Frederick Street
EDINBURGH
EH2 1EP
Tel: 031-225 4858
Fax: 031-220 3550

Real-time and database software specialists

WCSC PROFESSIONAL DEVELOPMENT TOOLS

DO NOT RE-INVENT THE WHEEL.

COMM-DRV-Professional Serial Communication libraries	$89.95 US.
COMM-LOG-True background XMODEM/YMODEM file transfer tool	$89.95.
MTASK-True preemptive multitasking kernel for MS-DOS	$89.95.
SCRN-INPUT-Character screen management package	$89.95.

TEL 713-498-4832
FAX 713-568-3334
ADDRESS: Willies Computer Software Co.
2470 S. Dalry Ashford Ste 188
Houston, TX 77077

DO YOU NEED MORE MEMORY?

We have three libraries to give you just that!

XMEM - Virtual memory Manager using base, Expanded, Extended or Disk Replaces malloc **£145**

MEMORY TOOLKIT - Library of routines to manage Expanded and Extended memory plus big DOS arrays **£95**

EMPTY SHELL - C routines for freeing up memory when shelling to a large program Uses EMS, XMS or disk to swap **£165**

GREAT WESTERN INSTRUMENTS LTD

Unit 1m Farrington Fields
Farrington Gurney
Bristol BS18 5UU
Tel 0761 452116
Fax 0761 453226
VAT Extra Carriage Free

Circle No. 865

Version Control

PVCS*	£345
MKS RCS	£155
SMS	£385
Sourcerers App	£169/399
TLib	£95

Editors

PVCS Editor (SPE)*	£175
QEdit	£45
Brief	£189
Epsilon	£169

Makes

PVCS Make*	£149
MKS Make	£95
Opus Make	£89

Others

SourceDoc*	£199
PolyLibrarian	£175
Demo II	£175
386/BlueMAX*	£49.90/£69.90
Project Director*	£495
SegMentor*	£1695

Call for site licences.

Readmar Systems
LIMITED

brainy
SYSTEMS (UK) LTD

Motherboards

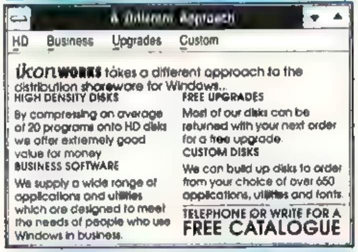
386SX 25	£110
386SX 25 32K Cache	£135
386DX 33 128K Cache	£235
486DX 33 256K Cache	£475
486-33 EISA	£930
MULTI I/O (IDE 2S 1P 1G	£14

Systems

WE CAN BUILD TO ANY SPECIFICATION PLEASE CALL FOR PRICES

Tel: 071 734 5783
Fax: 071 734 7256

windows
SHAREWARE



ikonworks takes a different approach to the distribution shareware for Windows..

HIGH DENSITY DISKS

By compressing on average of 20 programs onto HD disks we offer extremely good value for money

BUSINESS SOFTWARE

We supply a wide range of applications and utilities which are designed to meet the needs of people who use Windows in business.

FREE UPGRADES

Most of our disks can be returned with your next order for a free upgrade.

CUSTOM DISKS

We can build up disks to order from your choice of over 650 applications, utilities and fonts.

TELEPHONE OR WRITE FOR A
FREE CATALOGUE

ikonworks
22 Hope Street
[illegible]
[illegible]
[illegible] 524 39283

Soft As it Gets
Ed
the programmer's editor

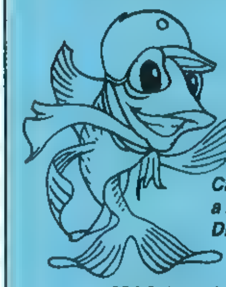
QBS Software Limited,

10 Barley Mow Passage,
London W4 4PH

Tel: 081-994 4842
Fax: 081-994 3441
BBS: 081-747 1979

© Soft As It Gets 1991

The Babelfish Paradox Data Driver for Clipper Summer '87 and 5.0x versions.



Call Now for a free Demo Disk.

QBS Software Ltd
10 Barley Mow Passage, London W4 4PH
Tel: 081 994 4842
Fax: 081 994 3441

BUGS WANTED!

A new BASIC User Group is looking for volunteer regional co-ordinators to help arrange regular meetings around the country.

If you have an interest or any experience with BASIC programming, please write to:

BUG
30 Osprey Close,
West Drayton, Middx UB7 7JE

PC Data Security Guide

Data Protection and Security for Personal Computers is a guide for support staff and managers. Everything you need to know about:

- Keeping viruses out of your company.
- Drafting and implementing a corporate data **security policy.**
- Persuading staff to take regular **backups** that are known to be reliable.
- Detecting and eradicating **software piracy.**

Includes comprehensive product guide and lots of trouble-shooting advice.

The price of £145 includes 90 days' **free telephone support** from the author.

Please call TTK Technical Publications on 081 995 9845 for details and a **free brochure.**

HALF PRICE!
PERSONAL DEVELOPERS KIT

TLIB Version Control
Opus Make
SourceDoc

Combined Price Usually £453

Our Price
£225

Readmar Systems
071 625 5255

The Secret of Successful Advertising is...

CLASSIFIED

To find out how you can reach 17,000 professional developers for as little as £85, call Marc Warren on 081 994 6477

To advertise in this section, please call Marc Warren on 081-994 6477. Fax 081-994 1533

More about Standards

Francis Glassborow explains the relevance of standards to users.

'ANSI' has become a familiar 'word' recently, while 'ISO' (International Organisation for Standardisation) is notable by its rarity. ANSI C took at least a year longer to reach its final form because of the needs of ISO. We very nearly had different standards between America and the rest of the world. Fortunately that was avoided, and ANSI has conceded all future responsibility for maintaining the C Standard to ISO.

While stocks last, you can buy a copy of the ANSI C standard, complete with rationale, for about £35. The ISO version (with different page numbers and without the rationale) will cost you about four times as much. You can buy either of them from the BSI.

If you believe the adverts and naïve reviews, almost all current C compilers comply with the ISO standard. Many are getting close, but few have conformance certificates. Among the C compilers for IBM PC compatibles running MS-DOS, only JPI's TopSpeed C has a conformance certificate, which it gained in early Autumn 1990. The command line compiler in the recently released Borland C++ 3.0 failed BSI's conformance test because it missed giving a warning in one instance. At time of writing, Microsoft has failed to pass with C/C++ 7.0. I am further concerned by the claim that the latter will compile code rejected by other compilers.

American companies usually base their claim to conformance on the Plum Hall suite. Their standards authority (ANSI) has not yet geared itself up to testing for conformance, so that is the best they can do on their side of the Atlantic. That is not enough for the UK.

The Plum Hall suite is a tool for compiler developers. Passing this test can never be sufficient, because there are requirements of the Standard that cannot be tested by a software suite. The document audit is one such item.

BSI has been testing compilers since Summer 1990, and the other European standards authorities have had procedures in place since then. It is past time for compiler vendors to get their advertising and marketing claims in line with the requirements of our trading standards.

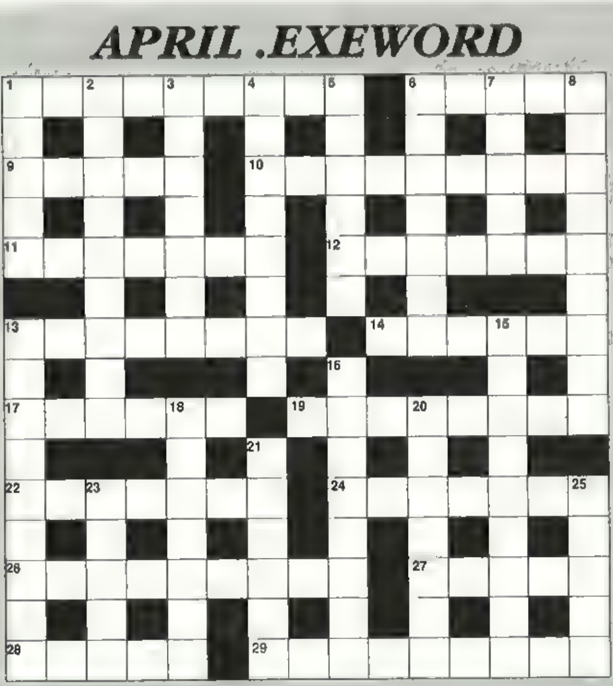
If you care about standards, then *Standard C* by Plauger and Brodie (ISBN 1 55615 158 6) and *The Standard C Library* by Plauger (ISBN 0 13 131509 9) are useful additions to your collection of reference books. The first of these is exceptional value at £6.95. At £29.95 the second is still excellent value, and contains a detailed commentary on the Standard C libraries with sample source code to implement most of the library functions. The source code is particularly instructive to those who want to write fully conforming C source code.

The ISO C standard is only one of several standards that, taken together, provide good support to those wanting to write portable code. Until POSIX is in place and supported by most programming platforms there will be several major barriers preventing you from writing portable code.

Consider the problem of direct processing of the keyboard. Just about all platforms give you a way of acquiring and processing raw keypresses, but the methods are not standard. The result is that there is no fully portable method for writing such code in Standard C. EXE

For information about CUG(UK) write to 64 Southfield Road, Oxford, OX4 1PA or ring 0865 246490.

APRIL .EXEWORD



ACROSS

1 US link (9)
6 Bit on the heap (5)
9 Strange behaviour of six-footer carrying a chip (5)
10 It adds waves to waves (9)
11 Noses observed at end of shift (7)
12 Lyre bit made to be free (7)
13 After Philadelphia I stank rather in the sub-continent (8)
14 Heavenly twins (6)
17 Demands instructions... (6)
19 ... those from the program suite (8)
22 It outputs inputs in a standard way (7)
24 Once I confused mist with big-headedness (7)
26 Finally reaching a total (9)
27 Links firmly but in a woolly way? (5)
28 French Russian (5)
29 Re the long view of development (9)

DOWN

1 Hardware failure in a bad day on the market (5)
2 Successful fishing when operating a modern system (9)
3 Goes above the limit (6)
4 OS/2 - pie in the sky? (4,4)
5 Chunk of manganese got when program chunk changed initially (5)
6 May vanish in water unlike some problems (7)
7 Up and down it goes, round and round (5)
8 Political propaganda on a shared circuit (5,4)
13 Method of working or where it happens (9)
15 Picturing an object round in free thinking ... (9)
16 ... boss in control of the limits (8)
18 Happily what aerials do (7)
20 Working bag (7)
21 Teaches set of packets ... (6)
23 ... and sits waiting on the line (5)
25 Sweet audio signal (5)

'.EXEWORD' compiled by Eric Deeson

D	A	T	A	■	P	R	O	C	E	S	S	I	N	G
B	■	A	■	S	■	U	■	U	■	A	■	T	■	R
M	O	N	I	T	O	R	■	R	E	L	I	E	V	O
S	■	G	■	A	■	A	■	A	■	A	■	M	■	U
■	P	E	R	T	■	L	I	B	R	A	R	I	A	N
C	■	N	■	I	■	L	■	L	■	M	■	S	■	D
E	N	T	R	O	P	Y	■	E	■	■	B	E	L	L
N	■	■	■	N	■	■	■	■	■	S	■	■	■	E
T	E	S	T	■	■	N	■	R	E	P	E	A	T	S
R	■	E	■	W	■	O	■	E	■	E	■	D	■	S
A	L	G	O	R	I	T	H	M	■	C	O	D	E	■
L	■	M	■	I	■	F	■	A	■	I	■	R	■	C
I	N	E	R	T	I	A	■	R	E	F	R	E	S	H
S	■	N	■	E	■	I	■	K	■	Y	■	S	■	I
E	N	T	E	R	P	R	I	S	E	■	A	S	A	P

MARCH .EXEWORD

SOFTWARE PROFESSIONALS

BERKS **£18k - £24k**
Development and support of **EIS object-oriented 4GL** applications using **GUI's** data manipulation and distributed data source connections (SQL, DDE, ASCII). You must be a high calibre graduate with excellent 3GL (eg. 'C') or 4GL (any) skills combined with strong communication skills.

BUCKS **£18k - £26k**
Revolutionary developer of emulation software products is seeking software and senior engineers with 2+ years in-depth experience of **'C'**, **UNIX** or **MAC** programming. Any **MS-DOS, Assembler, Windows, PC device driver** development, **Workstations** or **RISC** experience useful.

CITY **£20k - £25k + cheap mortgage!**
UNIX systems development, support and administration posts available if you have **UNIX** and **'C'** skills combined with some of the following: **C++, Ethernet, TCP/IP, LAN Manager** or **Sybase/RDBMS.** Very wide ranging job in this leading financial organisation so flexible attitude to work and team approach needed.

ITALY **to £25k**
Leading European Software House looking for Software Engineer with 1 years + **C++, OOD/GUI** development experience.

MIDDX **£ neg**
Developer of Project Management Software is seeking an experienced Analyst/Programmer to lead a team developing **GUI** toolkits and applications. Knowledge of **'C'**, writing **Widgets** combined with **UNIX** and **MS-DOS** is required. Any **Oracle RDBMS** and **SQL** desirable.

WILTS **£18k - £25k+**
Project Manager and Senior Software Engineer required to develop exciting new Office Automation product if you have **'C'** with **UNIX** plus either **MS-WINDOWS** development, **Microsoft SDK** or **Oracle** database development.

W.MIDLANDS **to £20k + OT**
Point of Sale and back office software development project requires software engineer with 4 years + experience of **'C'** under **MS-DOS.** Any of the following useful: **Requirements Definition, WAN's, HDLC, Novell, ArcNet, ISAM, Ctree,** or **Rtree.**

DUBLIN, EIRE **to £35k**
Foremost developer of PC applications software delivered on CD's is seeking Principle Engineer to port and develop future products eg **MOTIF** and **X-Windows.** You will need to be aged 28-40, 4+ years in depth **UNIX** and **'C'** with 2 years **Open Look** and/or **Motif.** Exposure to any of the following is useful: **C++, OO Techniques, Databases** or **4GL's.**

LINCS **£15k - £18k**
Design and development for bespoke software for publishing industry if you have **'C'/MS-DOS** with some of: **Graphics, GUI's, RDBMS, Comms, Networking** or **Assembler.**

W.Yorks **£15k - £20k**
Specialist producer of Spatial Information Systems incorporating maths modelling, relational databases and graphics is seeking software engineers with **'C'** and **Windows 3** experience.

G.I.S.

AVON **c £20k**
GIS Analyst/Programmer with minimum of 2 years **'C'** under **UNIX.** Some GIS tools experience useful.

OMAN **to £35k**
2 year contract if you have 2 years + experience of using **ARC/INFO** under **UNIX.**

For further information about these and MANY other opportunities either telephone
071 371-8496
or write to: ACUMEN SEARCH & SELECTION INTERNATIONAL
EBC House, 1a Ranelagh Gardens,
London SW6 3PA Fax: 071-371 8502 Evenings: 081 780 0636

ACUMEN

ASH ASSOCIATES

We specialise in the Recruitment of Software Design Engineers for the South East in Real Time Applications, Graphics, Communications, Control/Robotics, Signal/Image Processing & Modelling, Real Time Financial Systems/Databases.

OUR GUARANTEE

EMPLOYERS: Ash Associates offer you a personal service where we guarantee that an applicant we introduce to you is interested and is suitable for the position you are offering.

APPLICANTS: We will only put you forward for a position where we feel you have the right experience and will be of interest to you. In addition if we do not have the right position on our current files we will endeavour to find the right position for you.

DATABASE SYSTEMS DESIGN
SURREY £18K to £30K

Our clients seek experienced Database Support/ Developers including Project Leaders and Consultants.

These small but very successful companies offer exciting working environments, the opportunity to manage your own team and project with client contact, a realistic salary with benefits and for the more senior positions a company car.

You will be of Graduate Calibre or with several years experience including C, Cobol and/or Fortran plus Commercial Databases Oracle, Foxpro, Clipper gained in a PC (DOS) or Mainframe (UNIX) Environment.

FINANCIAL SYSTEMS
KENT £15K-£18K+Bens

This Major Financial Institution seeks to expand it's Software development team designing their New PC based systems.

You will use your C programming experience gained in a UNIX environment to develop Graphical and User Interfaces (OS Motif/X WINDOWS), Artificial Intelligent Transaction Systems and Computer Networks (Novell).

They will also be interested in any experience of the following, PC (DOS), NOVELL NETWORKING, WINDOWS 3.0, DATAEASE, SYBASE, PC FOCUS and analysis skills including LSDM/SSADM.

This opportunity is ideal for the Real Time Software Engineer seeking to advance his/her career into Financial Systems Development.

MEDICAL ELECTRONICS
KENT to £20K

Our client a small and very successful designer of Hand Held Diagnostic Systems seek a Highly experienced PC applications engineer.

You will be part of a small team but with sole responsibility for all PC Applications design including Graphics and User Interfaces, Databases and Data Aquisition Systems.

They need your C, Windows, Database (Lotus) and PC comms experience with your ability to take full responsibility for these projects and to lead a small team of software developers as they expand.

For these and many other interesting positions please call James Hunt or Ron Cook today and discuss your next career move.
TEL: (0425) 475480 (24Hrs) FAX: (0425) 480807

ASH associates
RECRUITMENT CONSULTANTS
3 Piper's Ash, Ringwood, Hants, BH24 1UF

ADVERTISERS INDEX

ADVERTISER	PRODUCT/SERVICE	CIRCLE	PAGE
Applied Logic	Object Oriented Programming	839	37
AI International	OOP Language & Environments	846	49
Bits per Second	Graphics for dBASE	835	34
BL Security	Security Dongle	857	75
Blink Inc.	Clipper Fast Linker	850	63
Borland	Borland C++	812	IFC
Borland	Borland C++	815	9
Brent Communications	MAX Copy Protection	861	80
BSO Tasking	Real Time Executive	830	26
Cocking & Drury	Smalltalk V	827	23
CTL	Copy Protection Hardware	838	37
Deepak Sareen	Project Management	842	41
DES	Software Protection	831	29
Evergreen	CASE Tools	847	50
Facts Direct	FAX Service	860	79
G-Force	Graphical Interface for Clipper	848	50
Glockenspiel	Class Constructors	825	18
Grey Matter	Programming Tools	813	3
GWI	Software Design Tools	820	15
Inst. Analysts/Progmrs	Institute	862	80
JPI I	C++ Compilers	821	15
JPI II	C++ Compilers	824	17
Lahey	Fortran Compiler	822	17
LPA	AI/KB/OOPS Software	819	15
Magnifeye	Software Protection Device	816	10
Microcosm	Copy Control	823	17
Microft	Security Software	852	66
Next	Workstations	828	25
Nu Mega	Debugging Tools	834	33
QA I	Evaluator/Software Testing	854	69
QA II	Training	864	OBC
QBS	Clipper Function Library	837	36
Richfords	Training	814	7
Roundhill	Development Tools	843	42
Salford Software	Fortran for DOS & UNIX	853	67
Select	CASE Tools	829	26
Silicon River I	Zinc/GUI Library	833	65
Silicon River II	Poet/Object Oriented Database	851	31
Softlok	Piracy Protection	856	72
Software Construction Co.	Programming Tools	826	22
Software Generation	Version Control	858	77
Software Paradise	Development Tools	849	56/57
Software Security	Security Products	818	13
Solution Systems	Programming Editors	841	39
Symantec	Zortech C++	845	47
System C	Application Generator	859	78
System Science	Programming Tools	855	71
System Star I	DOS Extender	817	11
System Star II	Software Tools	840	37
System Star III	Data Manager dbms	844	45
User Friendly	Software Copy Control	836	34

STOB - Beatification of Chernowell B

Channel 4's revealing investigative program 'Inside Chernowell B' raised many questions about that nuclear power plant's computer systems. Our contact on site, Parity Stob, has the answers.

Parity, could you tell us what the computers do at Chernowell?

Pretty well everything. We will provide a fully automatic system: control of the reactor and emergency shut-down, predicting and planning for surges in power demand, plus a really flash flexi-time system for the plant's personnel. There'll be no double-logging in for your mate while he's down the pub here, I can tell you.

I suppose security is pretty strict?

I should cocoa. Absolutely no ripped-off versions of *Leisure Suit Larry* here. We had a couple of viruses floating around the system a few months back, but now I'm 95% sure we've got them out of the system. We do run a virus-checker over the hard disks every now and then.

A virus! Are you sure the system is entirely safe?

Well, off the record, it would probably be sensible for the burghers of Chernowell to keep their washing inside next Friday the 13th.

How is the system put together?

We have a strict hierarchy of design documents, ranging from the BFP...

The 'BFP'?

The Big Fag Packet, which shows the design of the whole project, down to POTs -

- 'Packets Of Ten'?

You learn fast. Down to the POTs, which specify the action of each module. We did have a change control system based on liquorice Rizlas, but it all got stuck together in the hot weather.

But I'd hate you to get the wrong idea. We really are applying the principles of good programming practice right down the line. Including meaningful variable names. A quick glance at the Data Dictionary Fag Packet Global Booleans will give you the idea - look, there's `contamination`, `meltdown`, `evacuate`...

Quite. Moving on, your choice of programming language has proved controversial...

Well, we did consider using Ada at first, but we had to drop it.

Why was that?

For one thing we had enormous difficulty in getting anything to compile. We kept getting silly error messages: incompatible data types, illegal cast, you know the sort of thing. And then, even when we did get the code right, it used to take an hour for the compiler to run. We need to get this project out the door by 1993. We can't afford to have all our programmers sitting around drinking coffee all the time, you know.

So why did you go for BASIC?

We discovered that we could hire school leavers without expensive retraining. Given that we are using over 100 programmers on the project, this was no small consideration. We couldn't just take on graduates willy-nilly, you know.

Mind you, given the number of lines of code involved, I think the project leader is beginning to regret the choice.

Because of the lack of in-built safety features in the language?

No, because of the huge line numbers. Aside from the typing inconvenience, it's very difficult to remember if `GOSUB 101999764020` is *Shut steam valve B* or *Open steam valve B*. Especially since we had to strip out all the comments.

You what?

To get it to compile. The programs were just getting too large for the compiler; but then somebody found this excellent comment-stripping utility in the back of a 1983 *PCW* and we haven't had any trouble since.

Just one more thing. I hear you're leaving this job, selling your flat and moving to Newcastle - why is that?

Upwind.

RONICOM
RECRUITMENT

5-7 Sedley Place (off Oxford Street), London W1R 1HH
Tel: 071 491 3640 Fax: 071 499 2546

Comms/Email/Worldwide Network Specialist

Berks **£35k**

This is a corporate style Email system role for a top Software House. Your role will be to get the project up and running, you should be a graduate and have had notable successes in the past. Your experience will be not less than five years, and you should be self sufficient and not be frightened by the prospect of being a team of one! Suitable product experience could be UNIX MAil, Microsoft Mail or Courier. You must come from a coding background and be happy to get your hands dirty.

Cobol Consultant/Product Specialist

London **£Good**

Good development ability with RM Cobol, RM products, Microfocus Cobol or Microfocus products with either UNIX or PC platforms are required for this well known software consultancy. Interesting projects in many different client bases. You must have at least five years experience.

DIP Software Engineer

Home Counties **£Neg**

I have three software houses that are all looking for C/Windows programmers within either a UNIX or PC environment who are interested in, or have experience of Document Image Processing. I would expect you to be a graduate with at least three years coding behind you. This is a growing area within the IT Industry, I can therefore see good progression and stable tenure with all three companies.

Senior Software Engineer

Midlands **£18k**

You will be expected to produce software packages usually in assembly level language for incorporation into this hardware company's product range. You will be working by yourself or on small team projects. The kind of skills that are required are: good assembler (z80 and linker), MASM 8086 assembler and linker, MSDOS, and one high level language preferably C, Basic or Pascal. You must have at least three years experience and have a flexible approach to work.

Networking Technical Specialist

West London **£25k + Car**

CNE qualified, Novell guru, Ethernet LAN and WAN specialist? This major PC manufacturer is extremely interested in these kind of skills, they offer in return an interesting and varied job with lots of different projects all over Britain with some European work.

European Pre and Post Sales Technical Specialist

Berks **£20k + Car**

This company is involved in Client to Server connectivity: PC to AS400, to Mainframe etc. You must have good technical experience in 3270 comms and other protocols, have a knowledge of Windows and C, downsizing and emulation, and perhaps a use of Rumba and any other proprietary connectivity package. You should be well presented and be used to communicating with a range of Corporate clients up to fairly senior levels, some foreign travel is envisaged.

Oracle Ingres Gupta SQL

Home Counties **£25k**

Good development work with some of the most interesting database products available... This company is on the lookout for bright and enthusiastic software developers who are interested in working in a fast moving Software House. No recession here! Every successive financial quarter has shown increased profit and turnover, and as a consequence of this, your potential earnings are very good, and prospects greater...

Software Product Developer

London **£18k**

OOD, GUI, C, SQL your kind of thing? This innovative company is interested in the Mac approach to GUIs but on PCs under OS/2 with use of Application Manager and perhaps even Smalltalk.... It is a challenging role and they are looking for the final member of the team. You must be a high calibre Computer Science graduate with a good software engineering background behind you. There will be an equity deal involved after a qualifying period.

OSI C Windows Specialist

SW London **£Neg**

This well known OSI Software and Hardware interfacing development house is looking for experienced C developers with a good knowledge of comms protocols, and be prepared to be an instrumental player in future development strategies. In return there will be good benefits and bonuses as the recession has not affected this company......

MAC Database Software Developers

London **£above industry rates**

An innovative Mac Software House is looking for a cocktail of MAC development skills, (not DTP at all!) 4th dimension, Omnis, MPW, C and Mac App are the kind of areas of expertise required. You must have a good Computer Science Degree and be aware of recent programming techniques and developments within the Mac environment.

OSF Motif X-Windows UNIX developers

Cambs **£23k**

This renowned Company that is heavily involved with UNIX Windows and utility development is desperate for some extremely skilled individuals, you should have a good development background and be used to dealing with clients/end-users specific tailoring requirements. There should be some international travel involved, (User group and Conferences etc) and you should feel self sufficient in your own ability. Being a graduate is not necessarily a prerequisite, just being an able developer is....

OBJECT ORIENTED

Introducing the First Complete Education Programme in Object-Oriented Technology

If properly implemented the benefits of object technology to software developers can be immense, but as an emergent technology there are as many pitfalls as opportunities.

QA has a unique and unified education programme in Object-Oriented Technology (OOT) which will help you address the issues and deliver the benefits.

The cornerstone of our programme is the management-level *Strategic Briefing* explaining the benefits of OOT and how to evaluate, pilot and implement projects. An essential to all those who will be involved in OOT decision making.

Alternatively, you may prefer to begin with the *Primer* course, a technical introduction to OOT and an excellent first step toward both the *Design* and *Analysis* courses.

An integral part of our OOT programme is that for C++, one of the industry's most popular implementation languages to achieve your Object-Oriented goals.

OBJECT TECHNOLOGY STRATEGIC BRIEFING

OBJECT-ORIENTED PRIMER

OBJECT-ORIENTED ANALYSIS

OBJECT-ORIENTED DESIGN

C++ PROGRAMMING

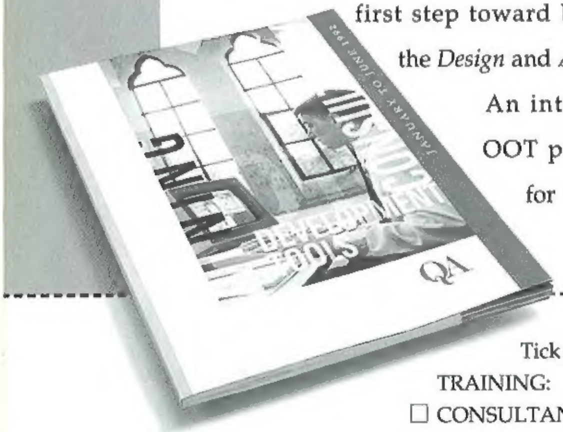
CONSULTANCY

Expert Consultation

As a result of our total commitment to OOT and the considerable expertise we have now developed in the subject, your company may prefer to explore and develop its commercial opportunities through our consultancy service.

You are invited to start your own chosen path to OOT mastery today:

To obtain full details of our carefully structured courses and consultancy in this new technology, which is set to revolutionise the world of software development, simply telephone Samantha Trinder on 0285 655888 or return the coupon.



Mail to QA Training Ltd, Cecily Hill Castle, Cirencester, Gloucestershire GL7 2EF, UK.

Tick here for details:

TRAINING: □ Object-Oriented □ Unix/Aix □ Networks □ OS/2 □ Windows □ C

□ CONSULTANCY

□ SOFTWARE DEVELOPMENT PRODUCTS

NAME ______________________ TITLE ______________________

.EXE 4/92

COMPANY ______________________

ADDRESS ______________________

______________________ TEL. NO. ______________________

QA TRAINING

First for Object-Oriented Education

AEXE1

CIRCLE NO. 864